Inside the Pentium FOOF Bug!

Dr. Dobb's JOURNAL

#285 MAY 1998

SOFTV
TOOLS
PROFESS
PROGRAMMER

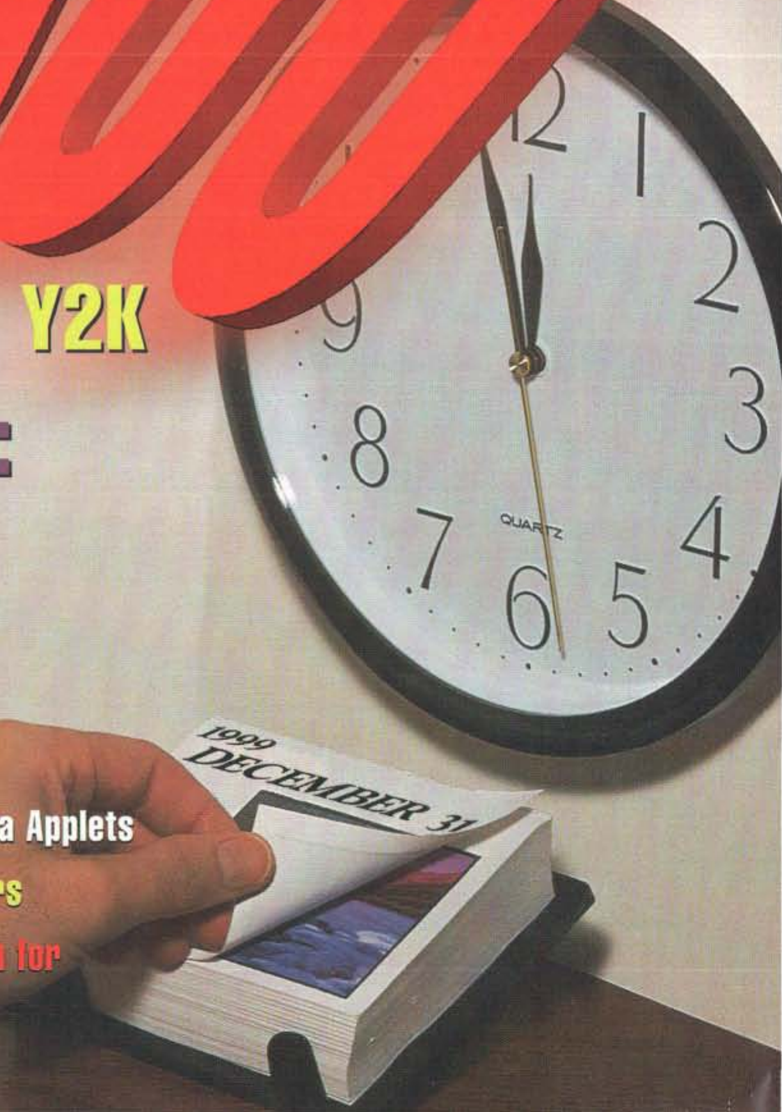
http://www.ddj.com

Swaine Meets Bob Bemer

NUMERICS and the Year 2000

Java Tools for the Y2K

Date Compression: A Y2K Solution?



Algorithms for High-Precision Numerics

The Hierarchical Data Format

Extending Windows CE Database Classes

Streaming Audio & Java Applets

Profile-Driven Compilers

Fast Memory Allocation for Embedded Systems

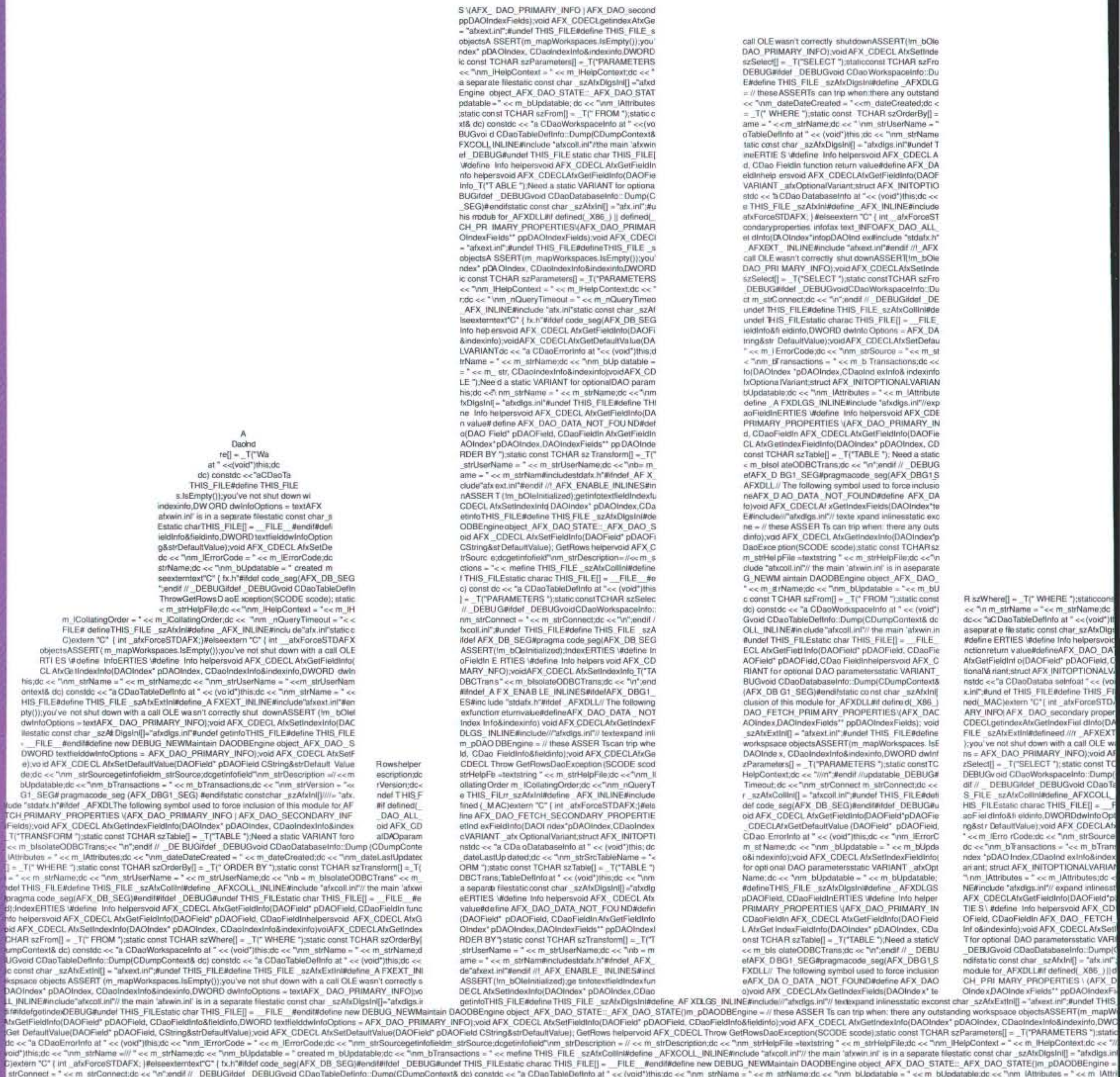
$4.95 ($5.95 CANADA)

05>
0 70992 35566 2

A Miller Freeman Publication

© 1997 Microsoft Corporation. All rights reserved. Microsoft, Visual C++, Visual Studio, and *Where do you want to go today?* are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and

Microsoft® Where do you want to go today?®



company names mentioned herein may be the trademarks of their respective owners.

Visual C++
Member of
Microsoft Visual Studio

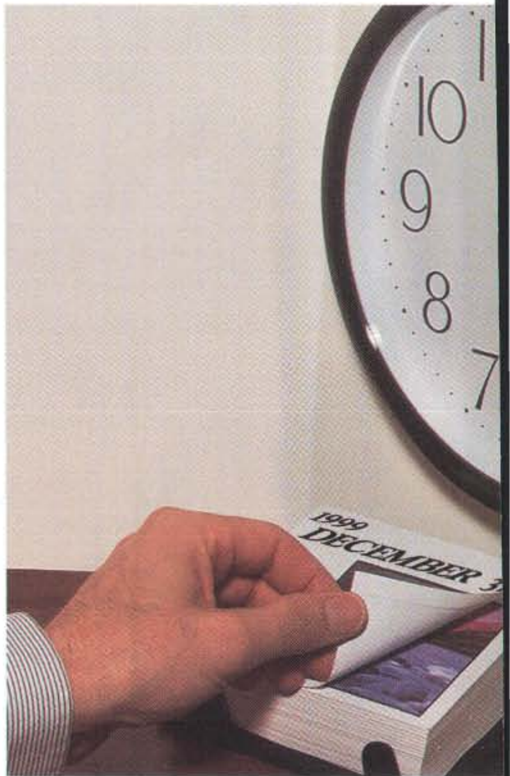
**It's data, not cement, that forms the foundations of big business.
And the leading Windows® NT-based database servers, from Oracle®, Sybase®, Informix®, and Microsoft® are all built in Microsoft® Visual C++®.**

www.microsoft.com/msdn/

Dr. Dobb's JOURNAL
SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

C O N T

FEATURES



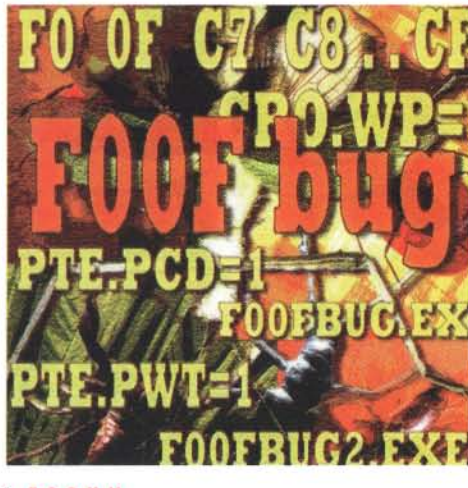
Cover photography by Pat Johnson Studios.

EMBEDDED SYSTEMS

INTERNET PROGRAMMING

ENTS

MAY 1998
VOLUME 23, ISSUE 5



PROGRAMMER'S TOOLCHEST

COLUMNS

FORUM

RESOURCE CENTER

As a service to our readers, source code (and related files), back-referenced articles, and relevant links are available electronically at this month's online Table of Contents at http://www.ddj.com/. Source code is also available via anonymous FTP from ftp.ddj.com (199.125.85.76), the *DDJ* Forum on CompuServe (type GO DDJ), and *DDJ* Online (650-358-8857, 14.4 kbps, 8-N-1). Source-code diskettes can be ordered ($14.95, California residents add sales tax) by mail, fax (650-358-9749), or phone (650-655-4100 x5701). Letters to the editor and article proposals/submissions should be mailed or faxed to the *DDJ* office or sent electronically to editors@ddj.com. Author guidelines are available at http://www.ddj.com/. Send inquiries or requests to *Dr. Dobb's Journal*, 411 Borel Ave., San Mateo, CA 94402. For subscription questions (including change of address), call 800-456-1215 (U.S. and Canada); other countries, call 303-678-0439 or fax 303-661-1885. E-mail subscription questions to 71572.341@compuserve.com or write to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188.

NEXT MONTH

Patterns and software design are our June focus, where we'll present tools for automatically detecting design patterns, examine the emerging discipline of "antipatterns," and more.

c-tree Plus®

COMMERCIAL Database Engine

Speed is essential in all database projects, but not at the expense of stability. You wouldn't try to go 100 miles per hour with your bicycle! The same is true in database technology. FairCom has been delivering fast, safe, full-featured database engines to the commercial marketplace for 19 years. Proven on large Unix servers and workstations, c-tree Plus's small footprint and exceptional performance has also made it the engine of choice for serious commercial developers on Windows and Mac. Check out www.faircom.com for detailed information. You'll be glad you did.

c-tree Plus® key features for $895:

- Royalty Free
- Portable Multi-Threaded API
- Complete C Source
- Thread Safe Libraries
- Standalone or Client/Server
- Complete Transaction Processing, including automatic recovery
- Save-points
- Abort/Commit
- Roll-forwards /Roll-backwards
- Easy make system
- Advanced Variable Length Records
- BLOBS
- Space Management
- File Level Security
- Conditional Index
- ODBC/Java Interfaces
- Over 25 Developer Servers included

Platforms:

MIPS ABI	DEC Alpha	Sun SPARC	Windows 95	SCO	Banyan VINES
88OPEN	OSF/1	DOS	Windows NT	Linux (Alpha/Sparc/Intel)	QNX
AIX	HP9000	OS/2	Windows 3.1	AT&T System V	Chorus
RS/6000	Sun OS	Mac	Interactive Unix	Netware NLM	Lynx

FairCom® Server

DEVELOPMENT SYSTEM

Half of your Client/Server project is the Server! You control 100% of your Client Side. Why settle for less on your Server side? Move your functions to the server-side to decrease network traffic and increase performance!

Today's database demands may often be too complex for traditional Relational Model Database Servers. Server needs come in many different sizes and shapes. What better way to accommodate these requirements than by allowing the developer to take full control of the Server side? FairCom's Server Development System was created to meet this need. It provides the developer the means to create an industrial strength Server. Complete make-files are included for all FairCom commercial platforms. With our proven kernel add or override existing database functionality or create your own special multi-threaded server:

Application Server	Network Gateway Server	Data Warehouse
Special Web Server	Departmental Database Server	Embedded Servers

FairCom Server Development System key features:

Provides complete source code for all the interface subsystems to the FairCom Server. Server mainline, Communication, Threading, Remote function interfaces and procedure calls are all supplied in complete C source code together with the FairCom Server sophisticated thread-safe kernel libraries.

Customizable	Rollback-Forward	Data History	Conditional Index
Transaction Processing	Anti-Deadlock Resolution	Multiple Protocols	Small Memory Footprint
Online Backup	Client Side Source	Heterogeneous Networking	OEM pricing
Disaster Recovery	Multi-threading	File Mirroring	ODBC/Java interface
			Key level locking

www.faircom.com

FairCom® corporation

Commercial Database Technology. Since 1979.

USA. 800.234.8180

Phone: USA 573.445.6833 · EUROPE +39.35.773.464 · JAPAN +81.0592.29.7504 · BRAZIL +55.14.224.1610

AD LINK 84

Dr. Dobb's JOURNAL — SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

PUBLISHER *Peter Westerman*

EDITORIAL

EDITOR-IN-CHIEF *Jonathan Erickson*
MANAGING EDITOR *Deirdre Blake*
MANAGING EDITOR, DIGITAL MEDIA *Kevin Carlson*
SENIOR TECHNICAL EDITOR *Tim Kientzle*
TECHNICAL EDITOR *Eugene Eric Kim*
SENIOR PRODUCTION EDITOR *Monica E. Berg*
ASSOCIATE MANAGING EDITOR *Amy W. Wong*
ART DIRECTOR *Margaret A. Anderson*
CONTRIBUTING EDITORS *Al Stevens, Tom Genereaux, Scot Wingo, George Shepherd, Robert Collins, Cliff Berg, David Betz, Bruce Schneier, Mark Russinovich, Bryce Cogswell, Ray Duncan, Jack Woehr, Jon Bentley, Dennis Shasha*
EDITOR-AT-LARGE *Michael Swaine*
PRODUCTION MANAGER *Denise Denis*

CIRCULATION

DIRECTOR OF CIRCULATION *Jerry Okabe*
GROUP CIRCULATION MANAGER *Michael Poplardo*

MARKETING/ADVERTISING

SALES DIRECTOR, EAST *Brenner Fuller*
SALES DIRECTOR, WEST *Paul Miller*
MARKETING MANAGER *Holly Vessichelli*
PUBLISHER'S ASSISTANT *Jodie Medeiros*
PRODUCTION COORDINATOR *Michael Calderon*
ACCOUNT MANAGERS see page 144
GRAPHIC DESIGNER *Carey Perez*

MILLER FREEMAN INC.

CEO *Tony Tillin*
CHAIRMAN *Marshall W. Freeman*
PRESIDENT *Donald A Pazour*
SENIOR VICE PRESIDENT/CFO *Warren "Andy" Ambrose*
SENIOR VICE PRESIDENTS *H. Ted Babr, David Nussbaum, Darrell Denny, Wini D. Ragus, Galen A. Poss*
VICE PRESIDENT/SOFTWARE DIVISION *Regina Starr Ridley*
VICE PRESIDENT/PRODUCTION *Andrew A. Mickus*
VICE PRESIDENT/CIRCULATION *Jerry Okabe*
GROUP PUBLISHER *Peter Hutchinson*

DR. DOBB'S JOURNAL **ISSN 1044-789X** is published monthly by Miller Freeman, Inc., 600 Harrison Street, San Francisco, CA 94107; 415-905-2200. Periodicals Postage Paid at San Francisco and at additional mailing offices.

ARTICLE SUBMISSIONS: Send manuscript and disk (with article, listings, and letter to the editor) to *DDJ* Submissions, 411 Borel Ave., San Mateo, CA 94402-3522.

DDJ **ON COMPUSERVE:** Type GO DDJ.

DDJ **ON THE INTERNET:** editors@ddj.com or http://www.ddj.com/.

SUBSCRIPTION: $34.95 for 1 year; $69.90 for 2 years. International orders must be prepaid. Payment may be made via Mastercard, Visa, or American Express; or via U.S. funds drawn on a U.S. bank. Canada and Mexico: $45.00 per year. All other foreign: $70.00 per year.

FOR SUBSCRIPTION QUESTIONS, change of address, and orders, call toll-free **800-456-1215** (U.S. and Canada). For all other countries, call **303-678-0439** or fax **303-661-1885**. On CompuServe, the customer-service number is 71572,341. Or write, *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188.

BACK ISSUES may be purchased for $9.00 per copy (includes shipping and handling). For issue availability, call **800-444-4881** in the U.S. and Canada. All other countries call **785-838-7500** or fax **785-841-2624**. Send e-mail to **orders@mfi.com.** Back issue orders must be prepaid. Please send payment to *Dr. Dobb's Journal*, 1601 W. 23rd Street, Suite 200, Lawrence, KS 66046-2700.

POSTMASTER: Send address changes to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80328-6188. **ISSN 1044-789X. GST (Canada) # R124771239.**

Canada Post International Publications Mail Product (Canadian Distribution) **Sales Agreement No. 0548677.**

FOREIGN NEWSSTAND DISTRIBUTOR: Worldwide Media Service Inc., 30 Montgomery St., Jersey City, NJ 07302; 212-332-7100.

Entire contents copyright © 1998 by Miller Freeman, Inc., unless otherwise noted on specific articles. All rights reserved.

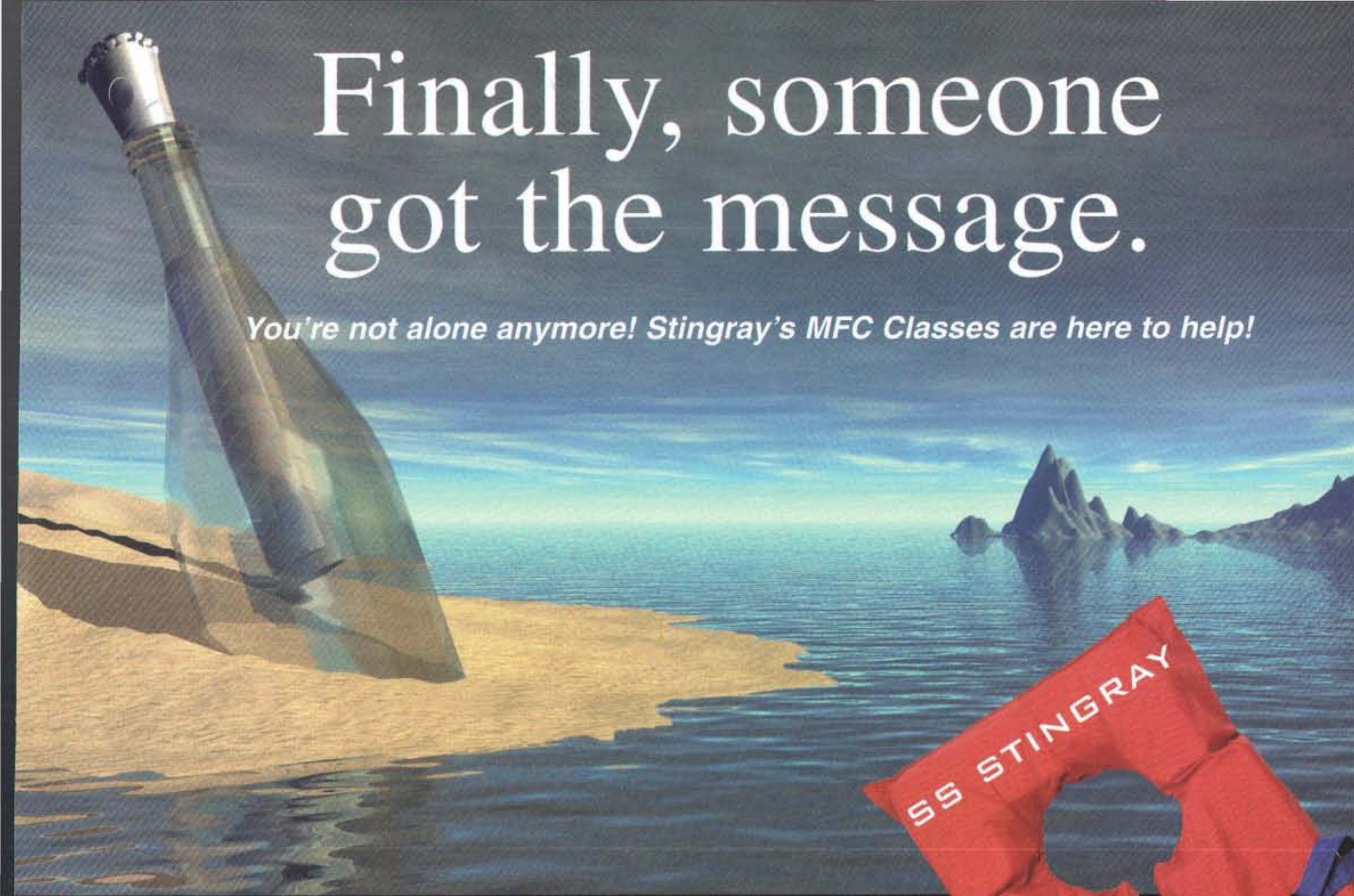
Estimated print run 171,000.

Miller Freeman
A United News & Media publicatior

ABP American Business Press

Printed in the USA

BPA



Finally, someone got the message.

You're not alone anymore! Stingray's MFC Classes are here to help!

Some times when developing software you can feel lost at sea. You know the great features you want in your applications, but are stranded due to limitations in MFC. *Stingray Software* can save you with a family of MFC extensions that add sophisticated GUI functionality in just minutes.

Have you ever tried to add docking windows, like Microsoft Developer Studio? What about shortcut bars like Microsoft Outlook? Or multi-selection tree controls? Do you want '97 "cool style" toolbars with drag and drop customization without spending months coding? **Objective Toolkit**™ provides these and over 70 extensions that add the coolest GUI functionality — in just a fraction of the time it would take you to build them from scratch. New **Objective Toolkit PRO**™ goes beyond Objective Toolkit to solve some of the more complex problems in the MFC architecture.

If you need grids and charts to present different views of your data — but the schedule is just too tight, then **Objective Grid**™ and **Objective Chart**™ are for you. You save hundreds of hours by using the built-in wizards and now Objective Grid has complete **Excel formula** support.

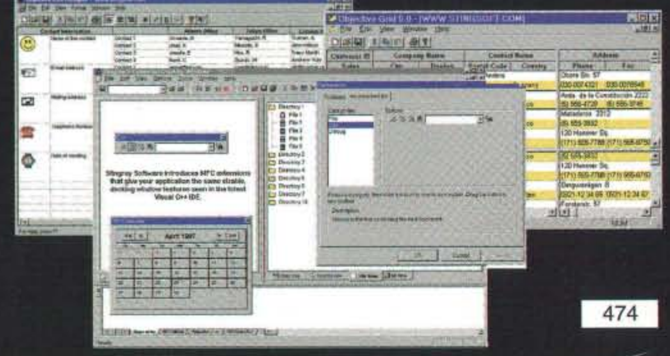
Do you want to provide graphical layouts in your applications? Don't spend months struggling with the GDI, use **Objective Diagram**™. It comes complete with zoom and print support plus OLE automation to transfer your layouts to other programs.

When you need to provide a full-featured color syntax editor, don't spend months writing code — turn to **Objective Edit**™.

With a family of over a dozen products, *Stingray Software* is fast becoming **the one-stop shop** for the object-oriented developer.

To find out how *Stingray Software* can rescue you, surf to our web site for **free demos, evaluation copies** and **white papers**.

All Stingray libraries come with FULL SOURCE CODE at NO ADDITIONAL CHARGE and have a 30 day money back guarantee.



474

Stingray Software, Inc. • 800-924-4223 • 919-461-0672 • email: sales@stingray.com

www.stingray.com

STINGRAY™ Software

The Next Generation of Development Tools™

All products and brand names are trademarks and/or registered trademarks of their respective holders.

AD LINK 296

Fear and Loathing on the Y2K Trail

Contrary to what you'll read elsewhere in this journal, the Year 2000 crisis has nothing to do with conversion, Cobol, or computers. No, the real problem is that the Year 2000 is an election year. And, even though the ballot boxes are more than two years from being stuffed, a rogue's gallery of crooks, con-men, and congressmen is already bellying up to the public trough.

For its part, the federal government showed again it's on top of things by recently establishing a Y2K task force to assist state and local governments, the private sector, and foreign governments. But no sooner had the government's press release hit the wires than the Information Technology Association of America (ITAA) weighed in to take credit, stating in its February 4, 1998, press release that "we suggested that a commission be impaneled to increase attention to solutions...to Year 2000 challenges." ITAA president Harris Miller went on to say that "the true count down to the Year 2000 begins today."

Sorry Harris, but in the real world, the Year 2000 countdown began a long time ago. In fact, according to those in the Y2K trenches, any yet-to-start project that will take more than 1.5 years to complete won't be finished in time. Underscoring this point, consulting firms specializing in Y2K conversion aren't taking on new projects that start after July 1998. And in some cases, 15, 20, or even 30 years isn't enough time to fix what soon will be broken.

Take global positioning systems (GPS), for instance. Over the past few years, millions of GPS receivers (not to be confused with GPS ground stations or satellite broadcasters) have been sold to sportsmen, geologists, archeologists, the military, taxi cab drivers, and others who want to know where in the world they are at any particular time. As part of the algorithm that lets the GPS receiver pinpoint location, GPS distributes time information. (A time differential is required to fine-tune rough triangulations when determining exact locations.) Since GPS time data is available throughout much of the world, other applications (such as financial computers) piggyback on GPS time/date information for a variety of purposes. Every millisecond, thousands of computers take time calibrations from GPS broadcasts for calculating interest on huge short-term electronic-funds transactions.

But in the GPS signal standard, dates use 13 bits to represent a time-unit offset from a conventional epoch date consisting of two fields (*epoch+offset*). GPS time receivers that have been programmed to update the *epoch* field will experience only a minor hiccup (if any). On the other hand, receivers that have this information burned into programmable read-only memory (PROMs) will likely fail because, on or about August 22, 1999, the date value will overflow this 13-bit type as satellites broadcast a new epoch. These hardcoded epoch time subsystems will think the calendar has been reset to the epoch in 1980. In short, any system that hardcodes the GPS epoch and is sensitive to the fact that "1980" is not "1999" will fail. Luckily for the vendors (but not users), most GPS time-receiver equipment is sold under a warranty that says "this equipment is not warranted to be suitable for any particular purpose."

This is only one type of problem faced by global positioning systems. There is, of course, the now-infamous literal value "19" in date types. There are also problems with other types, including type overflow problems at various dates throughout 1999; Y2K arithmetic that implicitly assumes dates later than December 31, 1999, are impossible; and implicit module-interface date-type conversions, to name a few.

Compounding the Y2K GPS problem is that it is difficult, if not darned near impossible, to construct valid GPS test cases to see what will happen at the Year 2000. Why? Because, say GPS experts, the future (time) states of the system depend on physical values (such as orbital elements and gravitational forces) that can only be accurately determined within about three months of the Year 2000.

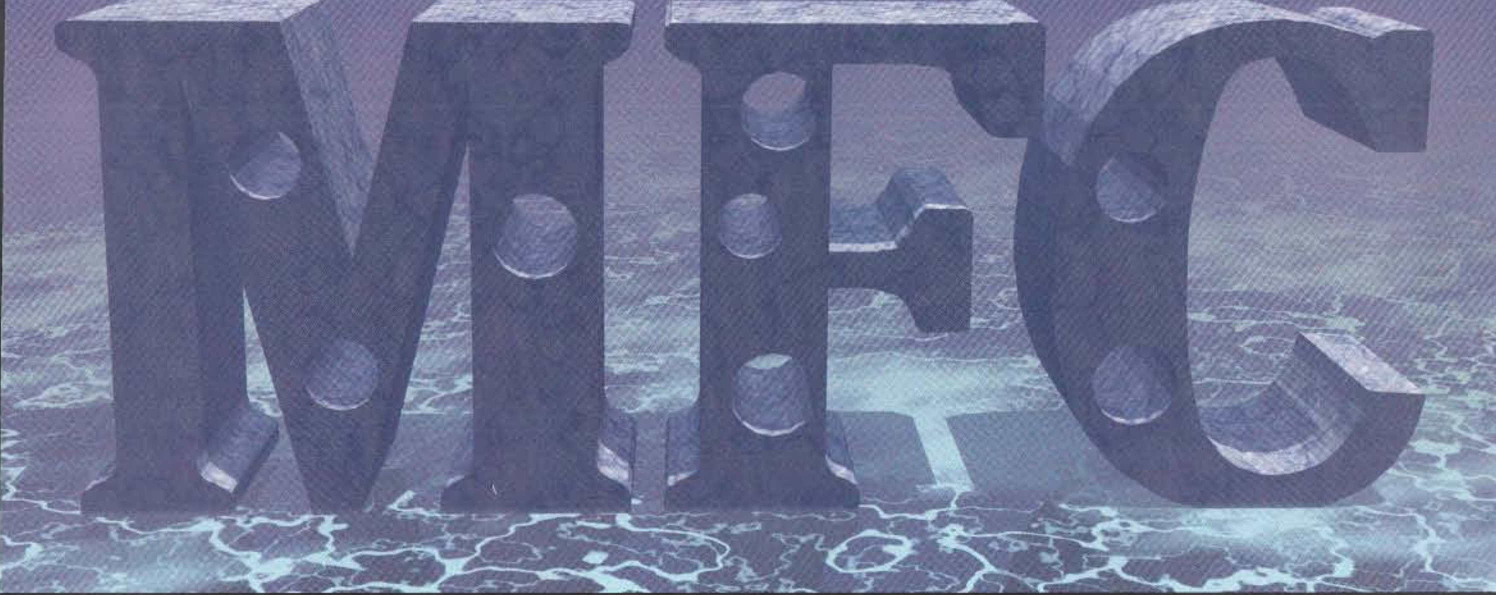
As outhouse luck would have it, the ensuing chaos will coincide with Year 2000 elections campaigns. In a more perfect world, errant GPS systems would deposit special-interest soft money into opponents' coffers, while routing candidates to hostile speaking engagements. Alas, the best we can hope for are candidates who advertise themselves as "Y2K compliant."

Jonathan Erickson

Jonathan Erickson
editor-in-chief
jerickson@ddj.com

Objective Toolkit™

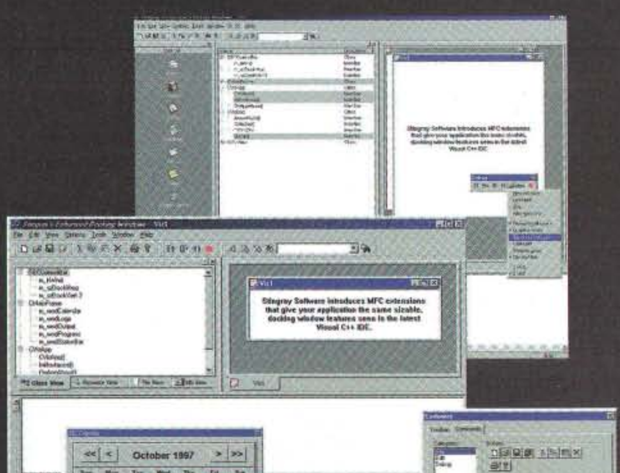
Filling the holes in MFC



The best MFC extensions money can buy

If you're a MFC programmer, you love it for its object-oriented design — but hate that most of today's coolest GUI features are not available. Features such as: docking windows like Microsoft Developer's Studio, '97 style cool-look dockable toolbars and menu bars with **full** customization, and shortcut bars like Microsoft Outlook.

For years Stingray Software has been filling the holes in MFC with **Objective Toolkit**. It contains over 60 extensions that add these and other great features such as: multi-column multi-selection tree controls, enhanced tabs and custom controls such as color wells, thumbnails, editable list boxes, calculator, calendar, masked edits, time/date and currency fields. To code these with MFC would take months, but with Objective Toolkit you're just minutes away from these sought after features.



Also check out our **Objective Toolkit Pro**, a popular add-on to Objective Toolkit, with the following advanced features: Layout Manager, dockable CViews, powerful scripting and forms engine, and a full model view controller framework — an alternative to document view.

With over a dozen products, Stingray Software is **the** object-oriented programming expert. Visit our web site for demos, evaluation copies, white papers or more information on this or any Stingray product.

All Stingray libraries come with FULL SOURCE CODE at NO ADDITIONAL CHARGE and have a 30-day money back guarantee.

Stingray Software, Inc. • 800-924-4223 • 919-461-0672 • email: sales@stingray.com

www.stingray.com/otmfc

STINGRAY™ Software

The Next Generation of Development Tools™

All products and brand names are trademarks and/or registered trademarks of their respective holders.

475

AD LINK 392

INTROD
A WHOLE N

Behold the Sun™ Ultra™ 5 and Ultra 10. Together, they represent a major evolutionary leap over all lower forms of workstations. Because they're the first full-fledged Sun workstations (with up to a 300MHz processor, no less) that'll let you run all your favorite PC apps. All while delivering the power, scalability, networkability, and proven robust UltraSPARC™/Solaris™ per-

*with 17" monitor

HOW THEY STACK UP IN ADVANTAGES

270/300MHz UltraSPARC IIi processor

Breakthrough Elite3D (Ultra 10) high-performance graphics

Up to 1GB ECC memory

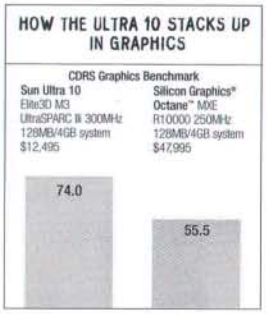
Industry standard dual PCI busses

Robust multithreaded Solaris operating environment

Runs PC productivity applications seamlessly (including Doom®)

Internet ready (Netscape™)

Java™ technology-enabled



formance you've come to expect from Sun. And all, believe it or not, starting at just $3,895. (Roughly 13% less than a comparably equipped Compaq® system.) With these workstations,

OF WORK

Pricing based on MSRP as of 2/98. CDRS as of 2/98. Certain models not available for Web purchase. ©1998 Sun Microsystems, Inc. All rights reserved. Sun, Sun
All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and other countries.

UCING
EW SPECIES

you can run heavy-duty applications one moment, then craft presentations using Microsoft® Office® the next. What's more, the Ultra 10 with its new Elite3D graphics will blow away a similar SGI machine. And at less than *a third* of the cost. And since both Sun models are binary compatible, they're perfect entry points to our full line of Sun systems (which, with up to 64 processors, can expand to meet anyone's needs). Both ready to run over 12,000 applications available for Sun. (And now every PC application beneath it.) All of which just may make them the missing links you've been looking for. For more information, call 800-SUN-FIND for a Sun reseller or representative near you. Or visit our Web site at sun.com/ult/ntc. THE NETWORK IS THE COMPUTER.™

Sun microsystems

STATION.

Microsystems, the Sun Logo, Solaris, Ultra and The Network Is The Computer are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Other trademarks are the property of their respective owners.

AD LINK 417

Resizable Data Structures

Dear *DDJ*,

In the January 1998 issue of *DDJ*, John Boyer discusses resizable data structures in "Algorithm Alley." Under the heading "Java and the STL," he discusses the Java hash table and the C++ STL *Vector* class.

I would like to add that Java in the java.util package also has a *Vector* class available. This Java *Vector* class is a fully resizable data structure. Its elements can be any object, including other vectors.

Using the constructor *Vector(0);*, this data structure uses array doubling as a capacity strategy, but any other strategy can be used by setting the *initialCapacity* and *capacityIncrement* parameters.

Using the *trimToSize* method of this class trims the capacity to the vector's current size, thus allowing for the *shrinkage* of the data structure that John is looking for.

More flexible shrinkage strategies, like shrinking the array to half size when it's one-quarter full, as John proposes, can be obtained by using *Vector's trimToSize* method followed by calling the *ensureCapacity* method with an appropriate argument. For the "one-quarter full" strategy, this argument would be two times the current vector size. This would look something like Example 1.

The availability of a fully resizable data structure (growing, shrinkage, different strategies) in Java puts the possibilities of this language in the right perspective.

Alexander G. Vandervoort
University of Groningen
a.g.van.der.voort@bdk.rug.nl

Dear *DDJ*,

In his excellent "Algorithm Alley" (*DDJ*, January 1998), John Boyer pointed out that resizable arrays should always be grown or shrunk by a factor of two. There are two additional comments which I would like to make:

- The factor which is used does not have to be two. The only important point is that the array size should be multiplied by a certain factor rather than increased by a fixed amount. One can choose a larger factor than two to improve execution times, or a smaller factor to reduce memory overhead. However, if a factor smaller than two is used, care should be taken to ensure that the array grows by at least one element when resized.
- This technique delivers *amortized* linear time complexity. This means that the average time required to insert an element is constant. However, the maximum time required still grows linearly with the number of elements. This may not be acceptable for some applications; however, in most cases, it should not be a problem.

Surprisingly, this powerful yet simple technique is very much underused. The number of programs that resize their arrays using fixed increments is staggering. As I found out recently, it even happens to the software giants. I wrote a Win32 program that tries to create as many windows as possible. Under Windows NT 4.0, the elapsed time increases quadratically with the number of windows created; it is fairly safe to conclude that this is caused by an array that is being grown by fixed amounts. (If anyone at Microsoft is reading this, please fix this in the next release.) Interestingly, the supposedly less powerful Windows 95 does not have this problem.

Martin Boehme
boehme@informatik.mu-luebeck.de

More Shunkworks

Dear *DDJ*,

It was interesting to see the letter from Quinn, Emanuel, Urquhart, & Oliver, LLP in the "Letters" section of *DDJ*, March 1998. To add to the lore, according to the *Trademark & Copyright Journal*, Vol. 55 No.1353 (pp. 82), Lockheed Martin sued Network Solutions Inc. (the domain name registrar) for issuing domain names such as "skunkworks.com," "skunkworks.net," "skunkwrks.com," and "skunkwerks.com." In a nutshell, "Lockheed sued NSI for trademark infringement, unfair competition, dilution, and contributory trademark infringement." Unfortunately for Lockheed, the "court granted NSI's motion for summary judgement on all claims."

Jeff Clausius
iometrics@iometrics.com

PTL Namespaces

Dear *DDJ*,

I really liked Al Stevens' column about PTL in *DDJ*, March 1998. I felt compelled to suggest that a namespace name that no one else would dream of using might be "AlStevensPTL" because if Microsoft or AT&T can do it, so can he. Besides, I like the way it looks and I always enjoy Al's articles. In my opinion, he deserves it.

John O. Viesselman, M.D.
doctorv@doctorv.com

Checking on VerCheck

Dear *DDJ*,

Although the VerCheck utility presented by John Graham-Cumming in *DDJ*, March 1998 is certainly very useful, I think a few, but important, comments are in order.

I also had to deal with this problem in the past and spent a few hours modifying Jeffrey Richter's well-known Vershow utility. I could notice a few problems with the way some programmers (especially at Microsoft) use VERSIONINFO. When you open the Properties dialog box in the Explorer by right clicking a filename, you get a Version tab that gives you the file version number. Like the VerCheck utility and unlike Vershow, this file version number is the "string" version number, not the "real" version number found in the VS_FIXEDFILEINFO data structure.

So what? Well, I have noticed that it happens more often than expected that the string version number doesn't match the actual (numerical) version number, especially for system files distributed by Microsoft. It seems that the developers there often forget to keep both entries in synch (this is done automatically in the the Visual Studio Resource Editor but not necessarily in all other development environments). So far as I know, installation utilities use the numerical version number, not the string value. At least, I hope so. Therefore, I think that any utility giving access to the file version number (including the Explorer itself), should also use the numerical file version number. Otherwise, this may lead to problems like replacing a newer file with an older file. Using the string version number is easier, but from far less reliable.

By the way, another problem with version stamping at Microsoft: The vast majority of the Win32 SDK include files don't contain any version information. I thought that Visual SourceSafe macros where able

```
if  (myVector.size()  >=  myVector.capacity() / 4) {
        myVector.trimToSize();
        myVector.ensureCapacity(2 * myVector.size());
}
```

Example 1: *Resizable data structures.*

Making software work together™

Making software work together - it's that simple. No matter how or where your systems are implemented, OrbixOTM will bring them together. Mainframe to Windows, Unix to Web - application integration has never been easier. Which allows you to focus on what really matters - using the best technology in the right place. And OrbixOTM lets you evolve your systems over time, adding new ideas to the reliable solutions that your business is built on.

But OrbixOTM does more than just integrate. It provides a complete enterprise solution. It features extensive and natural support for transactions, security, and systems management. There's advanced deployment support, load balancing and fault resilience. Which all means distributed object technology is now a long way from the lab. It's providing competitive advantage in the real world.

But then chances are your competition already knows that

Making software work together™

IONA Technologies

IONA Technologies at | **1-800 orbix4u** | **info@iona.com** | **www.iona.com**

Dublin Boston Perth San Mateo Hong Kong Frankfurt Washington DC London

'Orbix' is a registered trademark of IONA Technologies PLC. 'Making software work together' is a trademark of IONA Technologies PLC. All other products or services mentioned herein are trademarks of their respective owners. T3290

AD LINK 238

(continued from page 10)
to maintain this automatically. This is surprising. Any professional developer will add version information in the include file headers he writes, even for small programs. I can't imagine what good reason they have for not including version information in such important files. Since these include files may come with various products (Win32 SDK, VC++, Borland C++, and so on), it is often very difficult to determine which file is the latest and greatest when you install several of these products on your system. You absolutely *can't* rely on the time stamp. About two years ago (95/96), I received a new version of the Win32 SDK where all include files were dated 1993! Since then, I don't even look at the time stamp when comparing include files from Microsoft.

Patrick Philippot
patrick.philippot@iname.com

John responds: Thanks for your note, Patrick. My intent was to replace the long and arduous task of gathering the version numbers from Properties over the phone. I agree with your analysis concerning the string versus numeric version numbers and VerCheck could be improved to gather both quite easily. Also, I use the $History$ macro for automatic maintenance. This gives the full file history as well (as long as the developer bothers to write it).

Source Code Comparison

Dear *DDJ*,

In reading *DDJ*, February 1998, I noticed the "News & Views" item on MOSS, the program that provides a statistical comparison of source code, so that professors can detect cheating among programming students.

While I agree that this program could be useful, and that academic fraud should not be allowed, I know from personal experience that such similarities should not automatically be construed as fraud.

My freshman year in college, I was taking a Pascal course, and doing quite well. The day after submitting a project, I was summoned into an office with the Dean, my professor, and a classmate. The professor placed our respective projects side by side, and asked us to explain ourselves. Our code was virtually identical, with only minor differences in comments and variable names. We both admitted that the programs looked identical, and I think the Dean was ready to expel us both. Fortunately, we were both fairly meticulous notekeepers, and could fully document our development progress. Combined with an impromptu coding quiz in the Dean's office, we both managed to stay in school, although she and I were required never to be in the programming lab together or to send e-mail to one another, and we both got more than the average stares during tests. Eventually, I think the professor was fully convinced that our identical code was just a fluke.

For small projects, with programmers of similar training and experience, the likelihood of running across very similar code is higher than most people seem to think.

Byron Jones
byronj@fdpcorp.com

Yet Another Profiler

Dear *DDJ*,

Embedded in the Watcom C/C++ 11.0 compiler is a second profiler not mentioned by Ron van der Wal in his article "Source-Code Profilers for Win32" (*DDJ*, March 1998). When you compile the source code with the "-et" switch on the command line, the compiler inserts timing code at the start and exit of all functions, and it creates a LOG file for each program and DLL after each run. I have found this option to be more convenient and giving more accurate results than the stand-alone profiler that comes with Watcom.

Thiadmer
CompuPhase@compuserve.com

DDJ

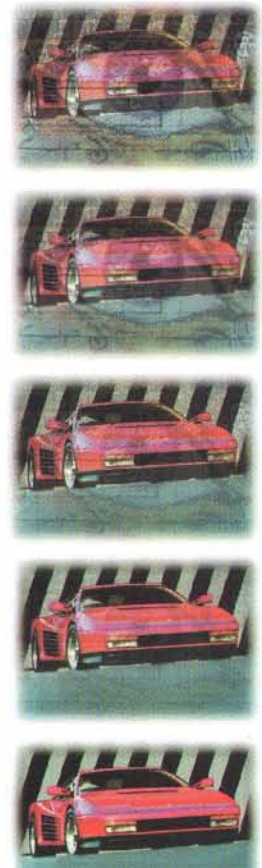
AD LINK 339

AD LINK 112

ADDING IMAGING?

not without

LEADTOOLS



MORE THAN 50+ FILE FORMATS

JPEG	WMF
DICOM	WPG
DIB	AVI
EPS	WAV
AVI	MIDI
ICO	MSP
CUR	VDA
DCX	GIF
IMG	-Embedded Text
IOCA	-Animated
MODCA	-Interlaced
FAX[raw]	-Transparency
WINFAX	TIFF 6.0
CALS raster	-MPT
GEM	-JPEG
PNG	-Packbits
PSD	-Huffman
MAC	-RLE
Photo CD (PCD)	-LZW
FlashPix (FPX)	-CMYK
Exif	-CCITT
TGA	-CCITT G31D
RAS	-CCITT G32D
PCT	-CCITT G4
CMP	-Bitonal
BMP	-Grayscale
OS2 BMP	-RGB
PCX	-YCbCr
AWD	-CIELAB

IMAGE PROCESSING

TRANSFORMS

Including resize, resample (interpolated resize), rotate (.01 degree), flip, invert, reverse, crop, underlay, shear, transpose, fill, auto deskew and combine bitmap (with mathematical and Boolean operations).

FILTERS

Including sharpen, blur, brighten, darken, invert, hue and saturation, intensity, contrast, gamma correction, histogram equalize, edge detect, line detect, emboss, mosaic, posterize, median and noise filters, spatial filter (which can be predefined such as gradient, laplacian, sobel, prewitt, shift and difference, line segment, or they can be customized), and more.

DRAWING

Draw directly to a bitmap surface using any of the windows GDI functions (such as TextOut, BitBlt, Ellipse, and Rectangle).

REGION OF INTEREST

Process only a specific portion of an image rather than the entire bitmap. Regions can be comprised of any combination of rectangles, ellipses, rounded-rectangles, freehand shapes, polygons, transparent color and more.

AD LINK 90

If you have an imaging project in front of you, choose the #1 toolkit in the market. LEADTOOLS is an award winning imaging toolkit that puts more than 7 years of development and millions of lines of code at your finger tips. You will save countless hours and have access to the best imaging technology available. Imaging technology that Microsoft, Hewlett Packard, Corel, Xerox and thousands of others have chosen for their products. With LEADTOOLS, your project is almost done!

VIEWING & SPECIAL EFFECTS

LEADTOOLS provides optimized rendering of any image to all display devices with monitor calibrations, auto-dithering, scaling, zooming, scrolling, and animation. Choose from 113 Paint Effects, 64 Dissolve Effects, and 24 Transition Effects, including pushes, pulls, wipes, splits, blinds, crushes, rolls, circulars, diagonals, stretches and many more with delays, grain sizes, pattern brushes (up to 64 passes), a colored wand and a transparent color. The combinations are unlimited. Other Special Effects include: 3D Shapes, 3D Rotated Text, 3D Frames, Gradient and Pattern Filled shapes and much more.

INTERNET/INTRANET

LEADTOOLS features a Net Aware ActiveX and a Netscape plug-in for Internet/Intranet applications, including a Bitmap Datapath allowing images to be read from any URL, Progressive JPEG, Progressive CMP, and support for GIF interlace, transparency, animation, and embedded text. A FeedLoad function has been created to allow image data to be displayed as it is being transmitted across the net.

DATABASE

LEADTOOLS has specific features designed for the imaging database developer: VB data binding, 32-bit ODBC, a customized OLE 2.0 in-place server, Load/Save memory, and Load/Save file offset.

LEADTOOLS is backed up by a 30-day money back guarntee (US & Canada only) and FREE technical support is available via phone, fax, Internet, CompuServe or BBS.

SCANNING

LEADTOOLS supports high speed scanning using ISIS and TWAIN. TWAIN includes both 16 & 32 bit native and buffered RAM transfer modes.

PRINTING

LEADTOOLS performs all image processing necessary to print directly to any Windows supported printer, with the ability to print text and multiple images on the same page.

COMPRESSION

LEADTOOLS offers more compression options than any other toolkit on the market, in both standard and proprietary formats.

THE POWER OF LEADTOOLS

LEADTOOLS is a collection of more than 400 functions, properties and methods that provide low level functions for complete control, all the way to the highest level functions for ease of use. Imaging Common Dialog Boxes, (new to v. 9.0) greatly simplifies integration. Royalty free and available as 16 & 32 bit DLLs, 16 & 32 bit ActiveXs, and a VBX and includes extensive source code examples for Microsoft Visual C/C++, Borland C/C++, Microsoft Visual Basic, Borland Delphi, Microsoft Visual FoxPro, Microsoft Access, VB and Java Script.

NOW FEATURING EASY TO USE LEADTOOLS IMAGING COMMON DIALOGS

"LEADTOOLS is an indispensable tool, we used it in the development of our FrontPage application".

Tom Button
Director of Marketing,
Internet Platform & Tools division
Microsoft

"The file format support is phenomenal, LEADTOOLS gave Micrografx's applications (like CreataCard™ and Picture Publisher™) the flexibility to import and export a wide variety of established and new file formats with ease. Their format implementations are complete and performance Great!"

Andy Cohen
Director Software Development
Consumer Products
Micrografx

CUSTOMIZE NOW

FlashPix™ MODULE

Integrate the full FlashPix experience.
- FlashPix viewing transforms.
- FlashPix non-image data.
- FlashPix thumbnails.
- Load and Save portions of an image.

CUSTOMIZE NOW

VIDEO MODULE

Integrate extended multimedia capabilities.
- Motion video editing.
- Record, Create, Edit, Play and Save AVI files.
- Record, Play, and save WAV files.
- Play Midi files.

ALSO AVAILABLE

PRO EXPRESS

Top of the line LEADTOOLS toolkit.
- Document Imaging.
- Annotation.
- Xerox TextBridge® OCR.
- Bi-tonal Filters.

Order now: 800-637-1837

900 Baxter St. Charlotte, NC 28204
704-332-5532 Fax: 704-372-8161 CompuServe:"GO LEADTECH"

LEAD TECHNOLOGIES INCORPORATED

LEADTOOLS is available in several versions, not all features are available in all versions. *License required from Unisys for formats using LZW compression. FlashPix Module and Pro Express require royalties. TextBridge® is a Registered Trademark of the Xerox Corporation. LEAD and LEADTOOLS are registered trademarks of LEAD Technologies, Inc. All other product names are trademarks of their respective owners.

Special Offers, Evaluations, Demos & more!

http://www.leadtools.com/

Programmer's Paradise®

the developer's *definitive* source for software!

The Hottest Development Tools! Order Now!

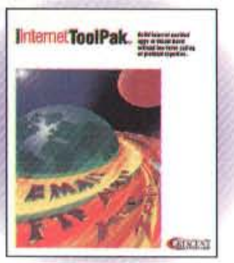
New Version

$99

Paradise No. L12 0E12-FT

ER*win*/Desktop Version Upgrade for Windows 95/NT

by Logic Works

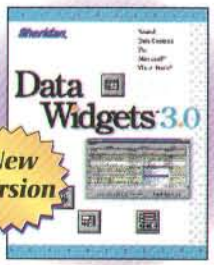


$235

Paradise No. P47 0330-FT

Internet ToolPak

by Crescent Division of Progress



New Version

$229

Paradise No. S16 0410-FT

Data Widgets 3.0

by Sheridan Software Systems

$349.95*

Paradise No. M47 0112-FT

MS Visual Studio 97 Pro Version Upg

by Microsoft Corporation

**Price after $100 manufacturer's rebate.*

WISE Installation System Enterprise Edition

by WISE Solutions

The WISE Installation System Enterprise Edition incorporates the award-winning WISE Installation System, and the tools required for any type of application deployment. The Enterprise Edition includes SmartPatch, WebDeploy, and SetupCapture. The WISE Installation System creates professional installation programs for Windows, Windows 95, and Windows NT. WISE is a completely Windows-based installation editing/testing environment.

Now Shipping New Version 6.0

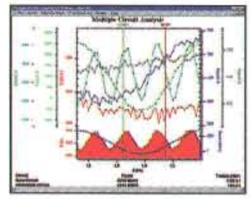
Paradise No. G10 0230-FT

$645

ProEssentials™

by GigaSoft®

16- and 32-bit DLL, OCX, VCL, and VBX interfaces providing charting functionality with consistent visual quality, real-world practicality, and overall professional appeal. Suited for engineering, financial, data-acquisition, and information system development. Be sure to find the best tool for your important development needs. Download a **demo** [500K] or get a **fully functional evaluation edition** [3Meg] at **www.pparadise.com/publishers/gigasoft**.



Paradise No. G04 0110-FT

$345

Visual SlickEdit

by MicroEdge

This award-winning editor increases development productivity, reduces costs of software maintenance and improves software quality through powerful features, software standardization and compatibility with your existing environment.

Software Standardization

With its multi-platform presence, integration with industry leading development environments and compatibility with version control systems, Visual SlickEdit provides your entire organization with a standard coding environment.

Paradise No. M39 0122-FT

$285

Sax Basic Engine Pro™

by Sax Software

Give your apps the power of a VBA-compatible macro language. Drop it onto your form and get started right away. The Professional Edition gives your apps a macro editor and debugger loaded with features. The Enterprise Edition also includes a high-performance server macro engine with multi-threading support.

New Version

Paradise No. S05 0310-FT

$455

VBTools 6

by BeCubed Software

Update your 16-bit applications today with any of the more than 60 custom 16-bit VBX controls! You'll save bunches of time with the new International control because it assigns internationalized text to all loaded controls! And there's a Flow Charting control, a 2D Slider, a Floating Text Extender and more, plus many other enhancements!

New Version

Paradise No. B30 0310-FT

$95

Ultimate Grid Studio Professional Edition

by Dundas Software

Here it is! Now you can get the Ultimate collection of grid components from one reliable source. Each component is designed to be the best in its class. Package includes Ultimate MFC, ActiveX, Java, and SDK grids. Components also available separately.

NEW!

Paradise No. D30 0210-FT

$549

VEDIT PLUS 5.1

by Greenview Data, Inc.

Quickly edit absolutely any text, data or binary files up to 2 Gigabytes in ASCII, Hexadecimal or EBCDIC. The new VEDIT PLUS for Windows/DOS does it all: database; mainframe; CD-ROM; Postscript; C; HTML; programming; word processing; translating; and converting. Exceptionally fast and compact. Fully configurable and programmable. A favorite since 1980.

New Version

Paradise No. G12 0110-FT

$135

Spread v2.5

by FarPoint Technologies

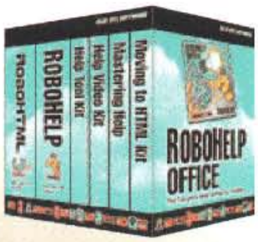
A complete spreadsheet control for most environments that support a VBX, OLE control, or DLL. Use as a spreadsheet to obtain variable lines of data or display tables of information. Data-aware—connect to databases with the Access Engine and ODBC. Includes over 250 properties and our improved Spread Designer. Formulas, sorting, and full-print support too.

Paradise No. F02 0110-FT

$229

Order online 24 hours a day, every day at www.pparadise.com

Visit www.pparadise.com for the full story on these and more great products! Use our product search button & order online 24 hours a day, everyday!



RoboHELP Office

Paradise No. B13 0231-FT

$665

The Fastest & Easiest Way to Create Help Systems

by Blue Sky® Software

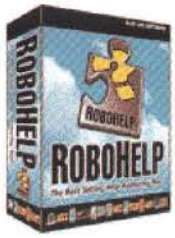
RoboHELP 5.5
The Best Selling Help Authoring Tool*

RoboHELP is the fastest and easiest way to create online Help for any platform. RoboHELP turns Microsoft Word into a full-featured authoring tool capable of creating professional Windows Help, WebHelp, Microsoft HTML Help, Netscape NetHelp, printed documentation, intranet/Internet Web sites, and Windows CE Help—all from a single source.

**Based on independent market survey.*

RoboHELP Office 5.5
The Complete Help Authoring Solution

RoboHELP Office provides a complete Help authoring solution with a collection of powerful Help and HTML tools for authoring professional Windows Help, WebHelp, Microsoft HTML Help, Netscape NetHelp, printed documentation, intranet/Internet Web sites, and Windows CE Help—all from a single source. Includes RoboHELP, RoboHTML (at no extra cost!), plus over 16 powerful Help authoring utilities.



RoboHELP

Paradise No. B13 0131-FT

$475

LEADTOOLS ActiveX Pro 32

by LEAD Technologies, Inc

LEADTOOLS offers technology in all major imaging categories for color, grayscale and bitonal imaging, with comprehensive functionality in each category. LEADTOOLS is an integrated development toolkit with more than 500 functions, properties and methods. Common Dialog boxes make LEADTOOLS easier to use and add-on modules are available for additional FlashPix, OCR, and Video support.

Paradise No. L05 0132-FT

$335

RoboHTML

by Blue Sky® Software

The Ultimate Way to Author HTML Help

RoboHTML is an HTML Help Authoring tool that provides a rich WYSIWYG editor with full drag and drop support, automated project management, and complete testing features. Unlike HTML editors, RoboHTML has built-in support for all the HTML Help features such as: Full drag and drop support for creating dynamic Table of Contents; multi-level Indexes; and Related Topics. Also supports easy creation of Popups, Navigation Buttons, Splash Screens, Shortcuts, Import existing WinHelp projects and more.

Paradise No. B13 0310-FT

$469

Doc-To-Help

by WexTech Systems

Create HTML Help, Windows Help for 95, NT 3.51/4.0 and 3.0/3.1, HTML, and printed documentation from one file, at the touch of a button, with Doc-To-Help 3. WexTech, the Microsoft acknowledged expert in HTML Help authoring, incorporates our experience in Doc-To-Help 3, enabling you to create great HTML Help.

Paradise No. WTD LP31-FT

$399

KEDIT for Windows 1.5

by Mansfield Software Group

KEDIT *for Windows*

Includes both 32-bit Windows 95/Windows NT and 16-bit Windows 3.1 modules. KEDIT is a powerful general purpose text editor with redefineable keys, undo/redo, selective line editing, regular expression support, enhanced syntax coloring, column oriented editing, file locking, a macro debugger, an IBM XEDIT-compatible command set, prefix area support, and more! The macro language is a subset of REXX. Also available in DOS and OS/2 text mode versions.

Paradise No. MAN KE31-FT

$135

Codewright Pro

by Premia

You'll work faster and easier when you use the right programmer's editor, Codewright. Now you'll browse code faster with Outline Symbols. It gathers information about your code in the background. You'll juggle changes with ease, with Difference Editing. It lets you selectively combine the changes from two revisions. The new Bookmarks Window lets you view bookmarks by name and by file. Now with synchronizing technology for Delphi and Visual C++ IDEs. With the help of the API Assistant, making complex function calls is as simple as filling in a form.

New Version

Paradise No. P36 0111-FT

$199

List Pro

by FarPoint Technologies

A set of customizable List Box and Combo Box controls. Create header groups and sub header groups and/or multiple columns to create a very unique list control. Merge cells in rows with same text to make display more user friendly. Bind to supported databases using Microsoft Jet Database Engine or Microsoft Access ODBC. Virtual memory manager allows connection to any size database. Supports single or multi-select lists, embedded icons and bitmaps in columns and rows, plus drag-and-drop capabilities.

Paradise No. F02 0211-FT

$189

in•no•vate

(in′ə vāt′) *vi.* to introduce new methods, etc.

Anyone can become an "innovator" using the software development tools you'll find in Programmer's Paradise catalog! We feature the best products from today's top publishers—at the best prices, of course!

Here's just a small sampling of what you'll find in the Programmer's Paradise catalog. To find all the tools you need to succeed—and innovate—call us today!

A Personal Guarantee!

"If you're not happy, we're not happy! You have my personal guarantee that we'll provide you with outstanding products, support and service."

–Roger Paradis
President & CEO
Programmer's Paradise, Inc.

ORDER NOW
or call for your free catalog!

1-800-445-7899

Call for shipping charges/return policy.
Prices subject to change without notice.

NASDAQ®: PROG

AD LINK 128

Developers: get the
at a price you can't

*Includes a limited license for development and testing only; BackOffice license limited to ten (10) simultaneous connections. ** Offer good through June 30, 1998. While supplies last. No shipping or handling costs.
© 1998 Microsoft Corporation. All rights reserved. Microsoft, BackOffice, Visual Basic, Visual C++, Visual Studio, Windows, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

power you need ignore.

For a limited time, get the power and stability of Microsoft® Windows NT® Workstation or BackOffice® Server 4.0 Developer Editions, free when you acquire a select Visual Tool.

Take your pick.

Choose the Professional Edition of the Microsoft Visual Basic® version 5.0, Visual C++® version 5.0, or Visual Studio™ 97 development systems, and get Microsoft Windows NT Workstation 4.0 Developer Edition* free. That's right. The most powerful desktop operating system for developing business applications with the reliability and security of Windows NT. Absolutely free. No shipping, no handling — Free.

Or, if you're building enterprise solutions, choose the Enterprise Edition of one of these same Visual Tools and get BackOffice Server 4.0 Developer Edition*— free. This suite of application servers, including Microsoft Windows NT Server, Microsoft Exchange Server and SQL Server™ help you build comprehensive business-critical, client/server solutions.

Either way you get the high performance and stability of Windows NT, along with the most powerful, easy-to-use visual development tools for building client/server, Internet, Intranet, and Windows®-based solutions.

This offer is too good to ignore. But it's only good for a limited time**. So act now. Visit your nearest reseller or call (888) 229-0738 to order. Then go to www.microsoft.com/developer/offer to get further details and to find out how to redeem your free copy of Windows NT Workstation 4.0 or BackOffice Server 4.0. All redemption will be accessed via a coupon on the web.

Microsoft® BackOffice®

Microsoft® VISUAL Studio™

Visual Basic®
Visual C++®
Visual InterDev™
Visual J++™
Visual FoxPro®

www.microsoft.com/developer/offer Where do you want to go today?®

Microsoft®

Kudos for Free Software Pioneers

The Electronic Frontier Foundation has awarded its 1998 Pioneer Awards to free software advocates Richard Stallman and Linus Torvalds. ("Free software" refers to the freedom, not the price, to redistribute the software, either with or without changes, either gratis or for a fee.) Stallman founded the GNU project in 1984, which led to GNU, a UNIX-compatible operating system. He is also the principal author of GNU Emacs, GNU C Compiler, GNU Debugger GDB, and other GNU programs. Torvalds, a recipient of the 1995 "*Dr. Dobb's* Excellence in Programming Award," developed Linux, a free UNIX-like kernel, which is estimated to have over five million users.

—Jonathan Erickson

PSCs: Personal Supercomputers

Interested in owning a Cray supercomputer? Tony Cole is your man. Cole's company, MemoryBilia (San Mateo, CA) sells various pieces of Cray supercomputers for prices ranging from a few hundred dollars for memory boards encased in lucite to several thousand dollars for a Cray-1 chassis (for those in the market for a very expensive bench). Five years ago, Cole bought Lawrence Livermore Labs' Cray-1, which was about to be junked, for $10,101.01 (it originally cost $19 million). Two years later, Cole called Cray Supercomputers to try and track down more equipment. After persisting for two months, Cole finally got through to Seymour Cray himself, and managed to buy a Cray-2, Cray-3, and Cray-4 for $5000. Cray may have got the better of the deal, since $5000 was more than they would have received from the junkyard. Cole has since added a Cray-YMP and Cray-XMP to his collection, and is planning to expand his business to other computer relics such as core memory. More information is available at http://www.memorybilia.com/.

—Eugene Eric Kim

Smart Dialing

These days, cellular phones and other hand-held devices can perform every communications task imaginable: e-mail, voice-mail, paging, web browsing. But even with all this palm-sized power, entering information remains a serious problem. Rather than miniaturize existing input devices or experiment with handwriting or speech recognition, Tegic Communications decided to make existing input methods smarter. Using a 12-button telephone keypad with three letters per key, Tegic's T9 software takes a sequence of keypresses, and compares the possible variations of character sequences with a dictionary to determine the word the user meant to type. The algorithm is patent-pending, and Tegic Communications is currently licensing dictionaries in English, French, German, Spanish, Italian, and Korean. For more information, see http://www.tegic.com/.

—Eugene Eric Kim

Let it Snow...

If you didn't get enough of last winter's cold and snow, check out http://www.mageesci.com/Antarctic/, where National Science Foundation (NSF) researcher Tony Hansen is posting video clips of his "vacation" to the South Pole. Hansen's primary goal is to collect information regarding the degree of environmental pollution, specifically aerosol black carbon.

—Jonathan Erickson

Math for the Web

The World Wide Web Consortium (W3C) has announced the release of Mathematical Markup Language (MathML) as a W3C Proposed Recommendation. MathML is a low-level syntax for representing structured data (such as mathematics) and mathematical expressions over the Web. In other words, MathML is an XML-compliant markup language that describes the content and presentation of mathematical expressions. MathML provides two sets of markup tags: one for presenting notation of mathematical data in markup format, and the other for relaying the semantic meaning of mathematical expressions. For more information, see http://www.w3.org/Math/.

—Jonathan Erickson

The Taxman Changes

According to a recent IRS pronouncement—Statement of Position (SOP) 97-2, Software Revenue Recognition—software companies must further standardize financial reporting. This may result in changes in the way some companies recognize revenue. For example, some companies may be required to defer reporting revenue longer, which could negatively affect short-term earnings.

—Jonathan Erickson

Advances in Nanoelectromechanical Technology

Researchers at the University of Southern California's Laboratory for Molecular Robotics have used a uniquely programmed atomic force microscope as a robot to push gold particles 15 nanometers in size into precise locations on a poly lysine-coated mica surface, spelling out the letters "USC." The gold particles are about 500 times smaller than a red blood cell and comparable in size to some molecules. Ari Requicha, professor of computer science and electrical engineering/systems and leader of the USC team, says that "Control over the structure of matter at the atomic or molecular scale will undoubtedly trigger a major revolution in man-made artifacts." Applications for this type of technology (which is called "nanoelectromechanical systems," or NEMS for short) include cell repair, ultrastrong materials derived from molecularly perfect prototypes, or compact disc machines with a thousand times more capacity than current models. For more information, see http://www-lmr.usc.edu/~lmr/.

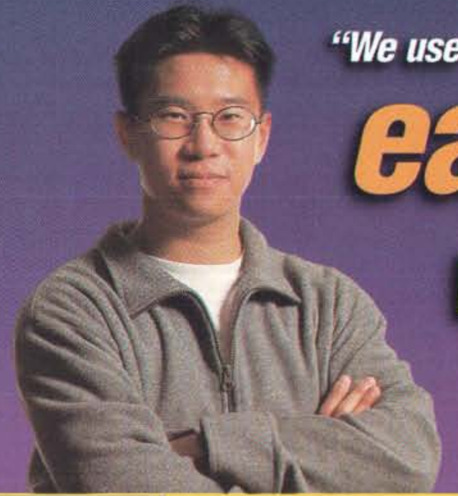
—Jonathan Erickson

Tcl Goes it Alone

John Ousterhout, the creator of Tcl/Tk, has left Sun Microsystems to found Scriptics, a company devoted to scripting tools, applications, and services. Scriptics' flagship product will be TclPro, a development environment for Tcl, which will include a source-level debugger and other tools. While no official release date has been set, Ousterhout expects the product to be ready by the fall.

Ousterhout noted, "We're really excited about this opportunity. We'll get to do a lot of things for Tcl that we've wanted to do for a long time." More information is available at http://www.scriptics.com/.

—Eugene Eric Kim



"We use KL Group components because they're

easy-to-use, well-supported & royalty-free"

Jorge Chang, Algorithmics Incorporated – makers of RiskWatch™

JavaBeans

JClass Java Beans – commercial quality, 100% Java components for JDK 1.0.2 or 1.1 let you unleash the power of your Java IDE.

JCLASS WORKS WITH:

- ✓ Java WorkShop
- ✓ JBuilder
- ✓ PowerJ
- ✓ SuperCede
- ✓ VisualAge for Java
- ✓ Visual Café
- ✓ Visual J++

and many more...

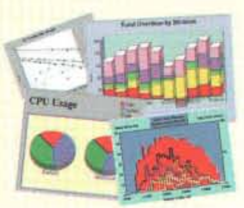
JClass BWT

JavaWorld Editors' Choice Finalist 1997

21 "must-have" Beans, including advanced tabs, outliner/tree and multi-column list

$49* BYTECODE

$199* SOURCE



JClass Chart

Feature-rich charting from the experts!

$399* BYTECODE

$999* SOURCE

JClass Field

Complete data input and validation, including calendar

$199* BYTECODE

$499* SOURCE

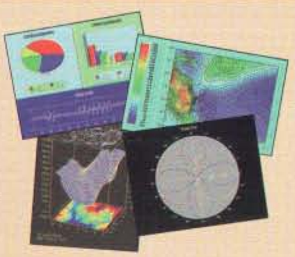
JClass LiveTable

The most flexible and powerful grid/table control available

$399* BYTECODE

$999* SOURCE

ActiveX



Olectra Chart

Speed, flexibility and control – charting power with no compromises. Includes 16- and 32-bit OCXs and DLLs.

$349*

Olectra Resizer

Easy, intelligent resizing behavior for Visual Basic forms.

$149*

UNIX Tools

1997 Editor's Award UNIXWorld

MOTIF WIDGETS

Discover why thousands of Motif developers find XRT widgets indispensable – call for your free 30-day evaluation.

UNIX PRINTING

PageFormatter – the easiest way to add PostScript printing to UNIX applications.

KL GROUP

Setting the standard for professional software development tools

KL Group Inc. Toll-Free: 1-800-663-4723 Tel: 416-594-1026 Fax: 416-594-1919

KL Group Europe B.V. Tel: +31 (0)20 679 95 03 Fax: +31 (0)20 470 03 26

www.klg.com

* Prices in U.S. dollars. Prices higher outside North America. JClass, JClass BWT, JClass Chart, JClass Field, JClass LiveTable, Olectra, Olectra Chart, Olectra Resizer, PageFormatter, XRT, XRT/3d, XRT/field, XRT/gear, XRT/graph and XRT/table are trademarks of KL Group Inc. All other products are trademarks or registered trademarks of their respective companies. DD001

Call 1-800-663-4723

Evaluate/Order **www.klg.com**

AD LINK 288

Date Compression and Year 2000 Challenges

Storage overflow is the problem

Robert L. Moore and D. Gregory Foley

Many systems with Year 2000 (Y2K) problems store dates in "MMDDYY" form. The first two places in this six-place field represent the month, the next two the day, and the final two the year. There are many different ways to encode MMDDYY information. Typically, each place is an integer type, or some other format (like characters), in the mechanics of the machine in question that represent the month, day, or year.

Six places can store a lot of information, but storage formats such as MMDDYY don't take advantage of this potential. For instance, the MM field only counts from 01 to 12, and the DD field only counts from 01 to 31. Neither field uses any of the remaining digits of storage potential. The MM field could (but does not) store the digits 13 through 99 (and 00). The DD field could (but does not) store the digits 32 through 99 (and 00). Potential storage space is lost because of the way information is recorded.

Ordinarily, not using the potential storage space available in six places doesn't matter. However, the field used to store the year is nearing the end of its storage capacity. The YY field will overflow when it goes from 1999 to 2000, because it counts from 00 to 99 (1900 to 1999 A.D., for instance). This overflow, in addition to other peculiar computer events that occur around January 1, 2000, (the use of "99" in the date field as a "save forever" indicator, the occurrence of a 400-year leap-year in 2000, the roll-over of the GPS time field from August 21 to August 22, 1999, to mention just a few) is at the heart of the the much-heralded Year 2000 problem.

Y2K is not just a software problem. It also impacts hardware, firmware, and networks. Y2K affects mainframe, client/server, desktop, and embedded systems. Neither is it just a Cobol problem. Applications written in Fortran, PL/1, C, C++, Perl, Ada, and the like are at risk for Y2K problems. The solutions we present here are from the viewpoint of C on a client/server system, but every comment is generally applicable. Ultimately, fixing Y2K problems is principally about fixing storage overflow.

Overflow eventually occurs in any storage format that records finite information. However, the MMDDYY date format overflows much earlier than necessary because of inefficient use of storage space in the MM and DD fields.

Y2K Solutions

A variety of solutions are available for fixing the overflow version of the Y2K problem. The two best known are expanding the MMDDYY date representation to MMDDYYYY (that is, using four digits for the year rather than two) or using a "logic" solution such as windowing (writing source code to interpret years into the correct century over a 100-year time span, for instance). Most Y2K repair discussions focus on these methods. (Another solution, applicable in limited circumstances, is to use the 28-year cycle of the Gregorian calendar's association between the day-of-the-week and the date in the years 1901 to 2099 inclusive. For example, July 4, 1998, is a Saturday as is July 4, 2026. We won't address that approach in this article.)

Date Expansion

Given enough time and resources, expanding date representations from MMDDYY to MMDDYYYY is usually the preferred method between expansion and windowing. Expansion offers several advantages:

- It lasts forever (well, from January 1, 0001, A.D. to December 31, 9999, A.D. anyway).
- For most people, four-digit year format is the natural representation.

Until recently, Y2K experts recommended four-digit expansion as the solution of choice. These recommendations have begun to change because only a short time remains to implement a Y2K fix. This brief remaining time span highlights the drawbacks to four-digit expansion:

Bob and Greg are senior software engineers at Coastal Research and Technology Inc. Bob is the author of a Year 2000 compliance criteria checklist which is widely disseminated throughout the U.S. intelligence community. Both Bob and Greg can be contacted at rlmoore0@erols.com.

- It must be implemented consistently everywhere dates are used or declared (source code, scripts, JCL, databases, computer-to-computer interfaces, user interfaces, and so on).
- It uses more storage space than two-digit year-date representations.

Changing from MMDDYY to MMDDYYYY is quite simple in abstract. In practice, you must find and appropriately alter all relevant declarations, output and input statements, calculations, and function calls. Furthermore, any variable that uses a changed variable (and any variable that uses the variable using the changed variable, and so on) must be checked and perhaps changed.

Windowing

One frequently suggested alternative to date expansion is a logic solution such as windowing. In windowing, dates of the form MMDDYY translate on-the-fly into unambiguous (MMDDYYYY, for instance) dates. Windowing is a simple concept. Human beings easily implement it mentally when we automatically translate a date like 07/04/98 to 07/04/1998, or 02/29/00 to 02/29/2000.

Windows come in two varieties—fixed and sliding. Fixed windows have a hard-coded comparison algorithm that prefixes a "19" to all two-digit dates in a given range (all two-digit year dates, where $50 < YY \leq 99$, for example) and a "20" to all remaining dates ($00 \leq YY \leq 50$). Sliding windows parameterize the comparison so that the range of dates moves as the current date changes; see Figure 1 and Listing One (listings begin on page 109).

Regardless of the windowing mechanism chosen, windows are quicker and require fewer coding changes to implement because they only apply where usage demands. Consider, for instance, the calculation AGE = CURRENT_DATE – BIRTH_DATE. For MMDDYY, this results in a negative (and hence erroneous) number if CURRENT_DATE and BIRTH_DATE fall on different sides of 01/01/2000. Windowing corrects this error. Windowing expands the date variable references to four digits at the point of calculation—AGE = WINDOW(CURRENT_DATE)–WINDOW(BIRTH_DATE). The declarations of AGE, CURRENT_DATE, BIRTH_DATE need not change, nor does any other usage necessarily need to change just because this usage needed a window. Reduced implementation effort is far and away the principal driver for windowing, although windows have the added advantage of avoiding the extra data storage space needed by date expansion.

Although windowing may be the solution of the hour, it is not without disadvantages. Windowing is a quick fix—it works for the moment. But don't count on it forever. The three major concerns with windows are:

- Their usable time span is no more than a century.
- They create additional testing challenges.
- They raise the risk that a future maintenance programmer may accidentally make an incorrect source-code change.

Sliding windows somewhat mitigate the first difficulty. They move forward as time advances, extending their life span (as opposed to fixed windows where the useful time span eventually expires). Still, windowing solutions are not appropriate in many places—for instance, the AGE calculation, if CURRENT_DATE and BIRTH_DATE are more than a century apart. Unfortunately, sliding windows suffer even more acutely from the second and third difficulties than fixed windows, because they are functionally more complex.

The two common solutions, expansion and windowing, neglect the fundamental cause of the Y2K problem—storage overflow. Computers can store much more information in six places than is allowed by the MMDDYY format. Storing more information in the same space is the essence of the Y2K solution

Photo: Pat Johnson Studios.

known as "compression." Nothing is really being "compressed." Compression simply uses a more concise date representation method. Compression is one of the more technically sophisticated Y2K solutions. Unfortunately, many programmers are not aware of it.

Compression

Compression shares many of the advantages and disadvantages of expansion and windowing. Among compression's advantages are:

- It can last for a very long time (for example, from January 1, 1600, A.D. to 768,151,959,528 A.D.).
- It does not require additional data storage space for expanded dates.

Compression also has comparative advantages with respect to windowing and expansion. For instance, compression requires no more source code changes than does expansion, yet it can store much more information. Further, except for output to human-readable forms, compression requires no function calls—a significant advantage over windowing. This does not mean that compression is suitable for every situation. Compression works well when:

- An application is built for the long term.
- The project to reach Y2K compliance is not a death march.
- Storage space concerns had some weight in design decisions.

The compression formats given later provide a wide range of alternatives for date storage. We've included C source code for converting between MMDDYYYY and each compressed format and back again. These calculations rely on knowing leap years; see Listing Two. These are useful when conversion to a user-readable format is required, or some calculation or interface with MMDDYYYY format is needed.

The compressed formats described next are in order of increasing date-storage capacity.

The CYYDDD Format

Using 1900 as a starting date (you can choose any date), one compression method is to count the number of years elapsed since 1900, and the number of days since the beginning of the year. The number of years is recorded in a three-digit field, where the first digit (C) represents the number of centuries elapsed since 1900 and the second and third digits (YY) are the year in the century. Places four through six (DDD) are the day count in the year beginning with January 1. For example, July 4, 1999, is 099185. July 4, 2055, would be 155185. This storage format can count 10 centuries, 99 years, and 365 (366 for a leap year) days; it can store a total of 1100 years. Converting MMDDYY to CYYDDD (see Examples 1 and 2) permits dates from January 1, 1900, to December 31, 2999.

This format is useful when some user-readability is important but storage space requirements do not permit date expansion, and date range requirements make the single-century-at-a-time limits of windows unfeasible. Expansion to four-digit years is usually preferable in this case, if practical.

The DDDDDD Format

Another approach is to count the number of days since a certain date rather than the number of years, months, and days. For example, storing date information as DDDDDD—counting days from January 1, 1900—means the same six places that can only store dates from January 1, 1900, to December 31, 1999, in MMDDYY can store 1 million days in DDDDDD. This is equivalent to more than 2700 years. Converting MMDDYY to DDDDDD (see Examples 3 and 4) lets you track dates from January 1, 1900, to past 4600 A.D.

This format is useful when storage space does not permit expansion and dating requirements do not allow windows. If possible, expansion to four-digit years is still preferable.

The MMDD 16-bit Year Format

One easily implemented and natural way of storing more date information is to leave the month (MM) and day (DD) fields alone and replace the year field (YY) with a bit register. The two-place year field is equivalent to a 16-bit long register, assuming that the computer in question puts eight bits in each character byte. Essentially, the format is the functional equivalent of MMDDYYYYY. (Note the extra Y.) Consequently, there is no need for a complicated algorithm—the year recorded is a YY expanded by windowing, just stored in binary rather than characters or digits. Since 16 bits stores $2^{16}-1$ = 65,535 years, dates from January 1, 0001, A.D. to December 31, 65,535, A.D. may be stored without ambiguity.

This format is generally more useful than four-digit date expansion, unless user-readability of machine-stored dates is a vital requirement. This format has a far wider range and uses less space than four-digit expansion, and converts to and from user-readable date information easily. The format also requires fewer function calls than windowing, hence is more efficient at run time. It also avoids windowing's date-span difficulties while using no more storage space than a windowed date.

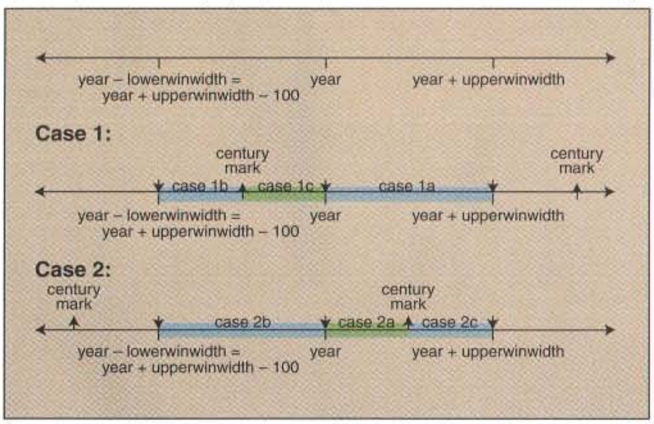


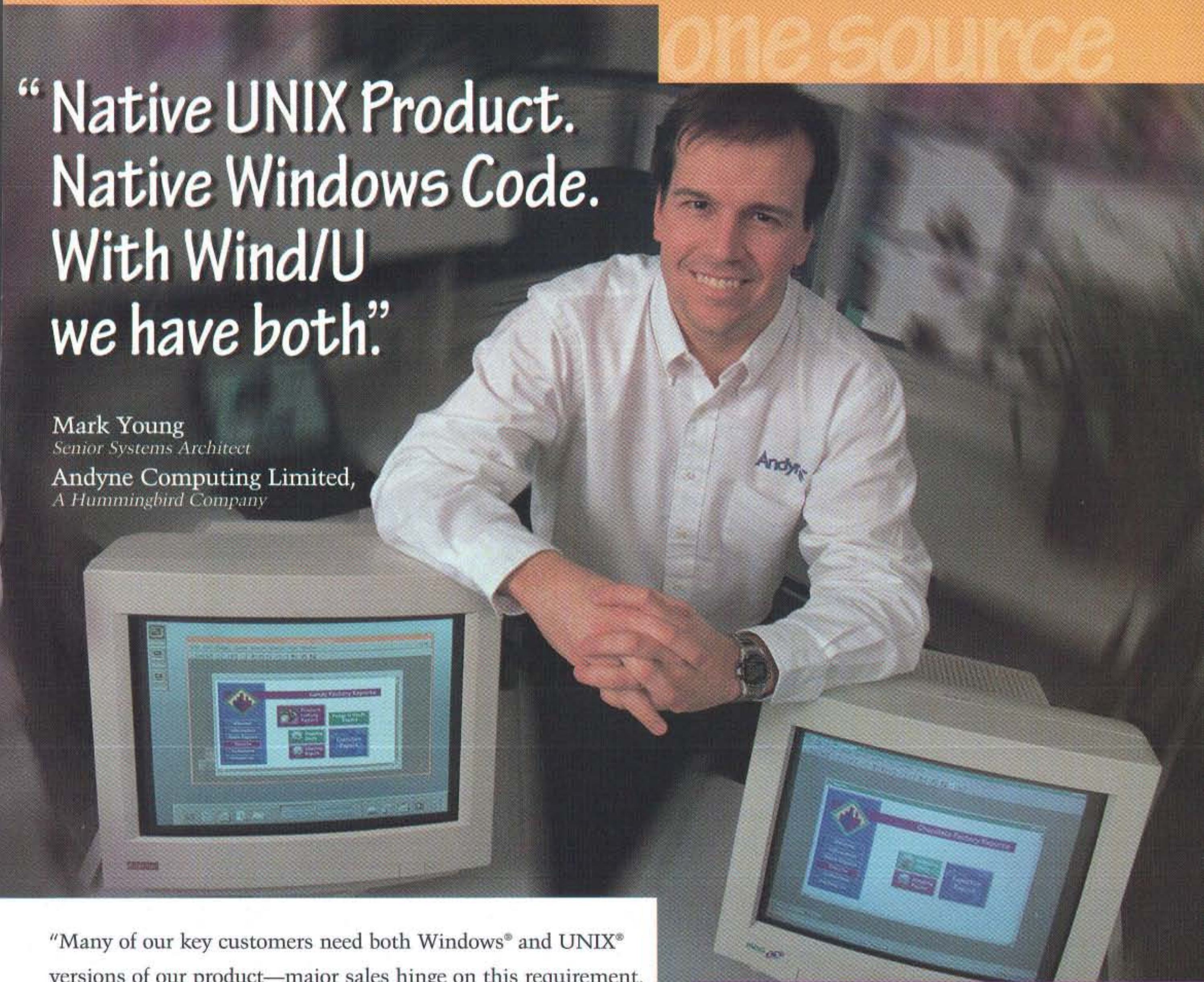
Figure 1: *Sliding windows.*

```
MMDDYYYY to CYYDDD

Represent YYYY as (Y1)(Y2)(Y3)(Y4). Then C =
(Y1*10+Y2)-19. The "YY" in the CYYDDD are Y3 and Y4.
Use MMDDtoDDD.c (Listing Three; available
electronically; see "Resource Center," page 3) to
complete the conversion.
```

Example 1: *MMDDYYYY to CYYDDD.*

```
CYYDDD to MMDDYYYY

Assuming the CYYDDD count begins with 1900 A.D.,
YYYY = (19+C)*100+YY. Use DDDtoMMDD.c (Listing Five;
available electronically) to complete the conversion.
```

Example 2: *CYYDDD to MMDDYYYY.*



"Many of our key customers need both Windows® and UNIX® versions of our product—major sales hinge on this requirement. Wind/U® is the only cross-platform solution that lets us use our Windows® source code to deliver a native UNIX version of Andyne® GQL®. With Wind/U, all the powerful Windows features that we use are available across platforms. And by using a single source, we **save the time and cost of developing twice** and get GQL to our UNIX/Motif customers fast.

Having Windows and UNIX versions gives us a competitive advantage. Wind/U helps us achieve this edge."

wind U®

- Win32 API & MFC on UNIX, OpenVMS, and OS/390
- The latest Visual C++ features
- Complete OLE, ActiveX, and COM
- Threads and WinInet APIs
- Native look-and-feel & performance
- Windows common controls & dialogs
- Industry standard online help & PostScript/PCL printing
- High performance OpenGL

Gain a competitive edge: www.bristol.com

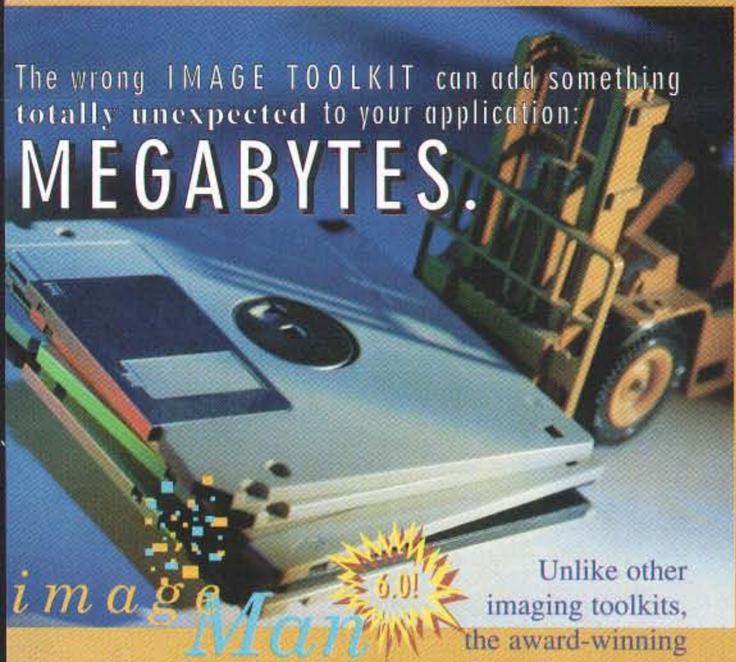
BRISTOL
T·E·C·H·N·O·L·O·G·Y
Delivering Cross-Platform Solutions

AD LINK 68

***Free White Paper*–"Delivering GQL on UNIX with One Source: The Wind/U Solution."(www.bristol.com/gql)**

In US: 203-798-1007 or in Europe: +31(0) 33 450 50 50

Wind/U and Bristol Technology are registered trademarks of Bristol Technology Inc. Andyne® and GQL® are registered trademarks of Andyne Computing Limited, which may be registered in certain jurisdictions. UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Ltd. All other products mentioned herein are trademarks of their respective holders.



Unlike other imaging toolkits, the award-winning **ImageMan** from Data Techniques, Inc. gives you all the features you need without adding instant code bloat.

At less than half the size of competing libraries, **ImageMan** is still the fastest and easiest way to add image support to your application. And our modular architecture means you can save even more space by including only the modules you need.

ActiveX
DLL
VBX

Our NEW version **6.0** sets the standard for speed and ease of use by introducing great new features like **special image drawing effects**, **interpolated scaling**, **image filters**, **enhanced color reduction**, and more. ImageMan also has support for over 30 image formats, including FlashPix, TIFF, JPEG, GIF, PhotoCD and PNG. And even our "old" features have been jazzed up, with improved scale-to-gray (antialiasing), easy-to-use dialogs, and more.

Visit our website to get the full lowdown on **ImageMan** – it's **guaranteed** to save you more than time.

ONLINE ORDERING
www.data-tech.com
ALL DAY EVERY DAY
email: sales@data-tech.com

Data Techniques inc

800-955-8015

838-682-4111 • Fax 838-682-0025 • BBS 838-682-4356

AD LINK 80

ImageMan is a registered trademark of Data Techniques, Inc.

```
MMDDYYYY to DDDDDD

Use MMDDToDDD.c (Listing Three; available
electronically) to convert the MMDD part of MMDDYYYY
to DDD--the day count within a year. Then use
DDDYYYYToDDDDDD.c (Listing Six; available
electronically) to convert DDD and the YYYY part of
MMDDYYYY to DDDDDD.
```

Example 3: *MMDDYYYY to DDDDDD.*

```
DDDDDD to MMDDYYYY

Use DDToDDD4Y.c (Listing Four; available
electronically) to convert from DDDDDD to DDDYYYY.
This is a four digit year and a count of days
starting at January 1 within that year. Use
DDDToMMDD.c (Listing Five; available electronically)
on the DDD part of DDDYYYY to convert DDD to MMDD.
The combined result is MMDDYYYY.
```

Example 4: *DDDDDD to MMDDYYYY.*

(continued from page 22)

The 48-bit Date Format

A format that allows storage of very long time spans is to count the number of days since a given date and store the information as a 48-bit long register. If the storage format of the computer being used equates eight bits to one byte, this 48-bit long register exactly replaces the MMDDYY date format with respect to storage space. This format is similar to the DDDDDD format. (Use the source code given with that format for conversions.) However, using bits rather than separate integers is a more efficient storage method. It is so effective that counting from January 1, 1600, A.D. allows date storage to 768,151,959,528 A.D. This is a sufficient time span for most applications.

The all-bits representation is useful when storing long date ranges. If desired, less than the full 48-bit representation may be used (say, 40- or 32-long bit register) to store date information with reduced storage space. For instance, using four eight-bit registers saves two bytes of space and is sufficient to store a count of 4,294,967,295 days, or about 11.75 million years. This format makes direct user interpretation of stored dates very difficult. Unless reduced storage space for dates is needed or long span date storage is desirable, the MMDD 16-bit year format is preferable.

Use the algorithms in Examples 3 and 4 (MMDDYYYY to DDDDDD and DDDDDD to MMDDYYYY) for conversions.

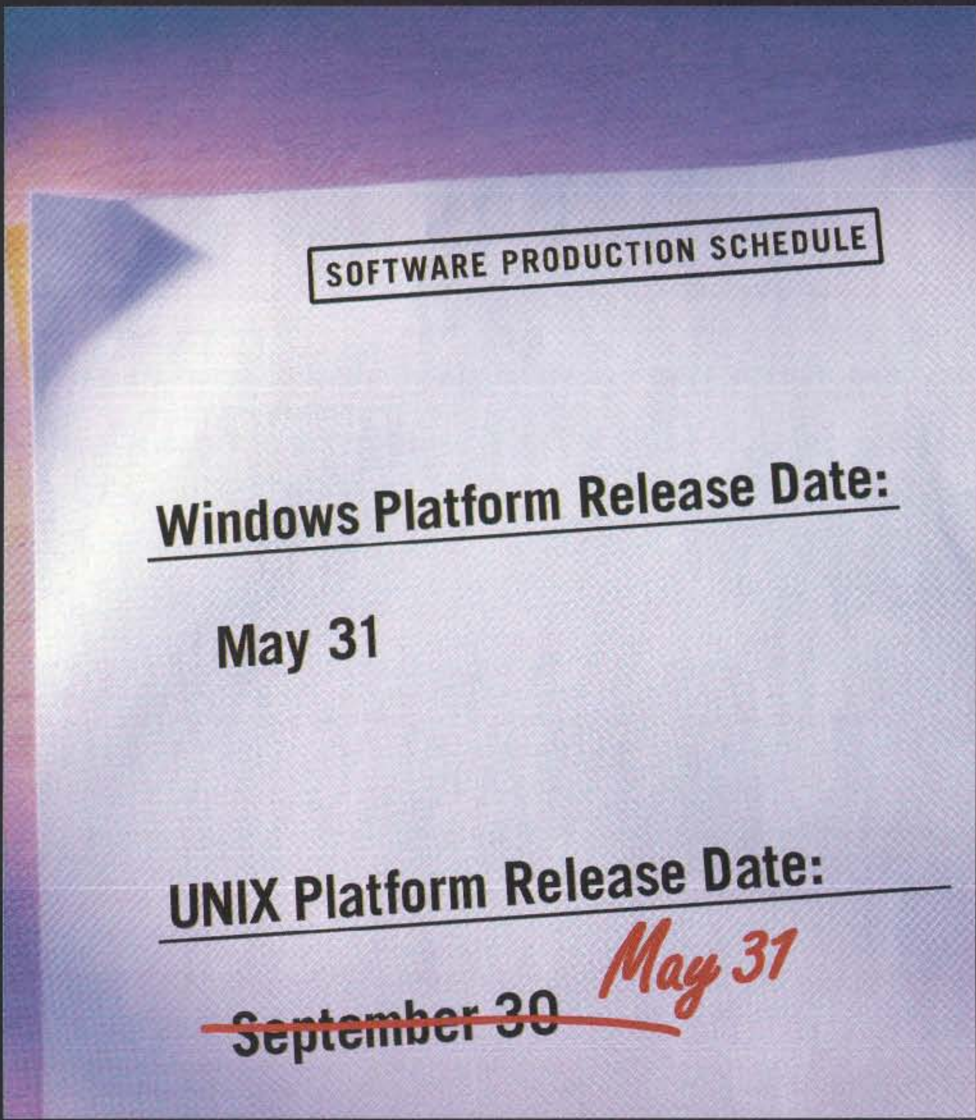
Conclusion

The Y2K literature suggests many other date storage formats. These formats are often variations of those suggested in this article (MMDDHH, counting years in hexadecimal rather than decimal digits or HDDCYY, counting months in hexadecimal and years as a century offset (C) plus years, and so on). The comparative advantages and disadvantages of these formats are similar to the earlier formats.

Date compression as a Y2K solution is not appropriate everywhere. For instance, windowing is the necessary choice of desperation for many, given the short time span in which we must implement a solution. Expansion is a straightforward choice, easily understood by anyone. However, when a project is not on a death march, when time spans longer than a century must be stored, or when using extra storage space is inconvenient, compression is an ideal Y2K solution.

DDJ

(Listings begin on page 109.)



Use MainWin To Port NT Code To UNIX, And You'll Only Have To Rewrite One Line.

Fast. Flawless. Flexible. When you use the MainWin cross-platform toolkit, you write apps for Windows once, then port the source code to UNIX, using UNIX-native WIN32 APIs. There's no need to double your programming efforts, no need to rewrite your code.

WWW.MAINSOFT.COM

Plus, you can still retain the same Test and QA staff, and even use the same documentation for both versions. The comprehensive MainWin environment ensures native performance to your existing UNIX customers, while speeding your own migration to the expanding world of Windows NT.

MAINSOFT

You will significantly reduce your development costs and simultaneously launch new applications on Windows and UNIX. For an evaluation copy of MainWin XDE, call us, send email to democopy@mainsoft.com, or visit our website. A few seconds with us might save you months of hard work.

1-800-MAINWIN

AD LINK 91

© 1997 Mainsoft Corporation, 1270 Oakmead Parkway, Suite 310, Sunnyvale, CA 94086. Telephone: (408) 774-3400, Fax: (408) 774-3404. MainWin is a registered trademark and Mainsoft is a trademark of Mainsoft Corporation. All other products or services names are trademarks or registered trademarks of their respective owners.

Strategies for Solving the Y2K Problem

Analyzing, converting, and testing legacy code

William Gothard and Les Rodner

The bulk of year 2000 work involves determining which applications must be fixed first, coordinating their repair with other internal and external systems, and testing repairs. This is a lot of work, and selecting the right conversion method can lead to significant time savings and affect the success of remediation efforts. In this article, we'll discuss the analysis, conversion, and testing processes, providing examples in Cobol.

Date Field Analysis

Before the year 2000 conversion process can begin, all date fields must be located in the application. Regardless of what type of remediation tool is being used, the single most important step in the entire year 2000 date impact analysis and conversion process is the creation of the character string or pattern list. This list contains all of the possible date field patterns in the application (*-DATE, *-YY, for instance) and literally drives the entire impact analysis and conversion process.

William is a senior development engineer at Intersolv specializing in year 2000 methodologies and remediation tools. Les is project manager/team leader for Intersolv Factory2000. They can be contacted at william_gothard@intersolv.com and les_rodner@intersolve.com, respectively.

Special care and consideration must be followed in the creation of the pattern list. If too many variations or combinations of date patterns are used, too many false positives (items marked as date fields that are not actually date fields) will crop up. If a pattern is inadvertently left out, the possibility exists for false negatives (missing a date field). The ultimate goal is to minimize false positives and eliminate false negatives. By testing sample code from the application, the pattern list can be fine-tuned to an acceptable tolerance of false positives/false negatives.

Missed date fields are difficult to identify during the initial search process. An essential step that aids in locating any missed dates is an analysis that occurs after completing the initial pattern search, a process known as "propagation" or "synonym" analysis. This technique discovers fields that do not have typical date field names, but are involved in date field actions (for example, MOVE WS-DATE TO WS-HOLD1). Chances are that the pattern search process would report on the entire line of code as a Y2K-impacted statement due entirely to the discovery of WS-DATE, but would not be capable of reporting on the line of code if WS-HOLD1 were then moved to another obscurely named field such as WS-HOLD2 (MOVE WS-HOLD1 TO WS-HOLD2, for example). Here we have two named date fields that would most likely be missed by the initial pattern-list search process. The potential for this type of situation requires that further analysis be done to discover any propagated or synonym fields. "Indirects" analysis independently searches for any parent/child/redefines/FD relationships in regard to the previously discovered date field, and adds any discovered date fields to the list of impacted items.

Another consideration when doing impact analysis is narrowing down the size of the fields to be searched. In Cobol, for instance, there are time stamps with lengths of up to 26 characters that contain potentially impacted dates. On the other end of the spectrum, fields with a length of one can usually be eliminated because they almost certainly have nothing to do with dates.

Date Conversion Methodologies

While there are several methods for handling date conversions, we'll focus on two — expansion and windowing. Expansion involves converting the physical field sizes to store the century value, thus achieving full four-digit compliance in both the application and data. This requires changes to both the file structures and all associated application programs.

In Listing One (listings begin on page 109), the two YY fields were identified for expansion in the BILLCOM-REC. Based on the decision to follow a date-expansion methodology, these fields must be expanded

Objects *r* Here.

Everyone agrees: the future belongs to objects.

But today's application developers have limited choices when it comes to harnessing the power of object technology without sacrificing performance, flexibility, or the freedom to use what you like.

Jasmine™ gives you the power you need with the freedom you want.

It's the first complete and pure object solution that has it all:

- The industry's easiest integrated development environment lets you drag and drop, and use all your favorite tools: native Java support, ActiveX controls, built-in VB integration, and C++.
- Jasmine is a powerful, true object database. It supports abstract classes, encapsulation, classification, inheritance (both single and multiple), unique object identity, methods (including instance-level, class-level, and collection-level), polymorphism, and aggregation.
- An industrial-strength architecture featuring SMP support, backup and restore, security, and transaction management.
- Built-in multimedia and Internet support: not only can you build and package the next generation of multimedia business applications; you can run them everywhere: Internet, intranet, extranet, client/server—all without recompiling.

Jasmine is real. A proven, complete object-oriented database and development environment. Unlike hybrid or partial object solutions, Jasmine actually works.

So you can shorten your time to market and gain a distinct competitive advantage.

Try it out—pick up the phone right now. Because Jasmine is ready today.

Are you?

Call 1-888-7JASMINE for your FREE Developer Edition CD or visit www.cai.com

COMPUTER® ASSOCIATES
Software superior by design.

Introducing Jasmine™
Objects @ Work™

©1998 Computer Associates International, Inc., Islandia, NY 11788-7000.
All other product names referenced herein are trademarks of their respective companies.

AD LINK 75

AD LINK 419

(continued from page 26)
to store the two-character century values. Listing Two is the revised code using expansion. When these two date fields are expanded from two to four bytes, it increases the overall size of the BILLCOM-RECORD from 100 to 104 bytes, thus creating the potential impact on all applications downstream that use the same BILLCOM-RECORD. Keep in mind that it is not a safe practice to subtract from any FILLER field, since another application could be using the FILLER area for a specific purpose.

With expansion of the Virtual Sequential Access Method (VSAM) file, Job Control Language (JCL) record lengths, blocking factors, and catalog parameters must be revised for all instances that use the expanded record. Expansion is a permanent solution. However, the level of effort and time required to analyze all interrelationships across applications—including the reorganization and restructuring of files, copy books, and databases—makes expansion a less viable option as the year 2000 approaches.

Even if a system has been fully converted using expansion, exposure still exists if the system is dependent on any external interface files coming in from other systems or companies. If those files have not been converted using the same expansion methodology that has been used in your applications, you must implement a bridging program that converts the incoming date to a format the converted target system can understand. Next, you must undergo the extremely risky task of deciding when to eliminate the bridging programs based on external system or interface file revisions.

The expansion scenario poses the greatest risk, especially this late in the game. Conversion takes the most time due to the interface analysis that must occur. It has the highest risk of failure due to errors of omission. Organizations with large applications that have not yet started the conversion process should not consider expansion, except for cases where expansion is the only possible choice.

Windowing

Windowing involves interpreting the century date field by inserting fixed or sliding windows into the application program logic. This enables programs to process two-digit dates without converting the files. Windowing based on a "pivot date" year (Year=30, for instance) is a reasonable approach that will:

- Achieve Y2K compliance in less time.
- Require less effort.
- Provide a targeted solution that will enable applications to function while plans are made to migrate the process to an environment that enables storing century values.

The windowing technique is fairly generic and can be adapted to any application by adjusting the pivot year date.

In Listing Three, a commitment must be made to retire the application's environment by the year 2029 or else rewindow the application as the year 2030 approaches. The rewindow effort may be more straightforward if the windowing code technique stores the pivot year literal as a field name in Working Storage and common routines are used to execute the window logic.

Suppose the value in POLICY-START-DATE is 290101 (January 1, 1929). In this case, Listing Three will conclude that the year is 2029. Now, suppose the value in POLICY-START-DATE is 300101 (January 1, 1930). In this case, Listing Three will conclude that the year is 1930.

Windowing's impact on data stored in files and databases varies depending on the application type (personnel, banking, insurance, telecommunications billing, and so on). All affected data stores must be thoroughly analyzed to identify those instances containing dates that will not fall within the 99 years that span the pivot year boundaries.

Windowing analysis is only concerned with cases such as

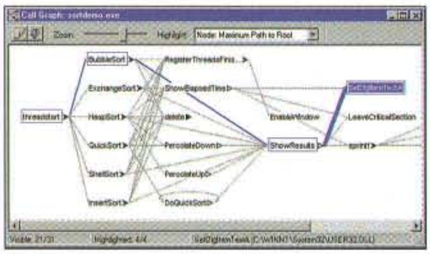
- Test date fields for greater than/less than conditions between fields and/or literals.
- Test date fields for equal conditions against date literals (equal tests between date fields are ignored).
- Subtracting a value from a year.
- Actions involving reference modification.
- Moving century literals.
- Existing window logic.

Windowing's primary advantage over expansion is that you don't have to worry about legacy data integrity other than finding the correct pivot date. But you still have to find every date-related manipulation function. For windowing to simplify matters when implementing a more permanent solution, care must be taken now in how the analysis results are documented. If done correctly, the window strategy will permit future rewindowing efforts to move the pivot year efficiently and in a timely manner. A rewindow effort would only involve revising the pivot year value stored in Working Storage, plus any unique workarounds documented during the original window implementation.

An effort to retire/rewrite the system should include a complete reengineering strategy to include all of the major system development life cycle (SDLC) phases

It's a lot easier to see where performance bottlenecks are when you can actually see your performance bottlenecks.

Let's face it: If you're looking to spot the source of most performance bottlenecks, you're spending an inordinate amount of time operating in the dark. Unless, of course, you're using Visual Quantify™ for your performance tuning. By clearly displaying performance data in graphical and text format, Visual Quantify will open your eyes to a whole new way of pinpointing the portions of code that slow down execution speed. With Rational's patented Object Code Insertion technology (OCI), the impact of source code, third-party components and system DLLs can all be measured, allowing developers to systematically eliminate slow run-time execution. Sound interesting? Just wait until you see it. If you're a Windows NT® developer using Visual Basic®, Visual C++® or Java™, visit our web site in order to download your free evaluation copy or call us today to find out more about Visual Quantify.



Visual Quantify lets you visually identify performance bottlenecks fast. See for yourself.

Visit us at www.rational.com/vquantify or call toll-free: 1-800-353-7873

RATIONAL
SOFTWARE CORPORATION

AD LINK 315

The word Rational and Rational's products are trademarks of Rational Software Corporation. References to other companies and their products use trademarks owned by the respective companies and are for reference purposes only. ©Copyright 1997 by Rational Software Corporation. Subject to change without notice.

Serious Copy Protection and Software Licensing with No Hardware Key or Disk Key

Distribute your software via Internet, CD or floppy with full security.

CrypKey provides solid copy protection with a broad range of features to control all aspects of your product's operation. You can offer free, automatic one-time trials, turn demos into full multi-user network versions, or sell specific options by phone, fax or email. Sell usage by time, runs, features - the control is all there - easy to implement, and even easier to support.

CrypKey also offers choices - integrate our SDK into your software, or use CrypKey Instant to protect your compiled exe in 5 minutes; no coding.

Try it free from our website to see why we're rated #1 by professional programmers who seek the best combination of security, features, ease of use, reliability & price.

100% Satisfaction Guaranteed!

CrypKey

http://www.crypkey.com

Kenonic Controls Ltd., Calgary, Canada Ph: 403-258-6200, Fx: 403-258-6201, email: info@crypkey.com

AD LINK 310

www.scarabsoftware.com

focus on the task at hand

You're a developer. Smart. Creative. Driven. You know all about deadlines. So why invest needless man-hours getting your software to install gracefully when you could outsource and save yourself time, money and headaches?

When you outsource your installer with Scarab you are freed to focus your in-house efforts on the job at hand, the job for which your team was assembled in the first place. Every Scarab installer is bound by an ironclad bug-free guarantee; less worry, more production. With a basic routine costing only $995, we will help your project stay on schedule and on budget.

toll free: 1-888-91-SCARAB

SCARAB

custom installers

get it right the first time™

AD LINK 420

(continued from page 28)

of requirement analysis, functional design, detail specifications, and so on. During this reengineering effort, existing logic may be determined obsolete and formulas may prove outdated, in addition to reports/screens that may require total revamping or elimination. The Year 2000

Missed date fields are difficult to identify during the initial search process

effort will provide an accurate inventory of all system components that will be useful as a starting point for reengineering activity.

Determining Pivot Date

The pivot date year selected depends on the application or business requirements. It is usually determined by the oldest date that is processed and is also affected by any "forward date" processing logic.

To adapt the data to the pivot year criteria, unique approaches are required. These include logic that recognizes specific instances that have out-of-bound dates and proceeds to execute unique logic for those instances. For example, records with the year of 100101 (January 1, 1910) would incorrectly be windowed as January 1, 2010. In this scenario, the pivot year may be lowered to accommodate the early date but would reduce the future life span of the window. This would create extra expense for the company to have the window pivot year changed again in the near future. Two options available to work around this case are: 1) Identify these exception records to the program (social security number, account number). When an exception record is encountered by the program, special logic is executed that windows and processes the affected dates based on a nonstandard pivot year. 2) Create a file to store these exception records and process these records in a separate program. This requires extracting these records from the file; creating an identically formatted file to store only these exception records; selecting a pivot year based on the "oldest"

The World isn't Flat,

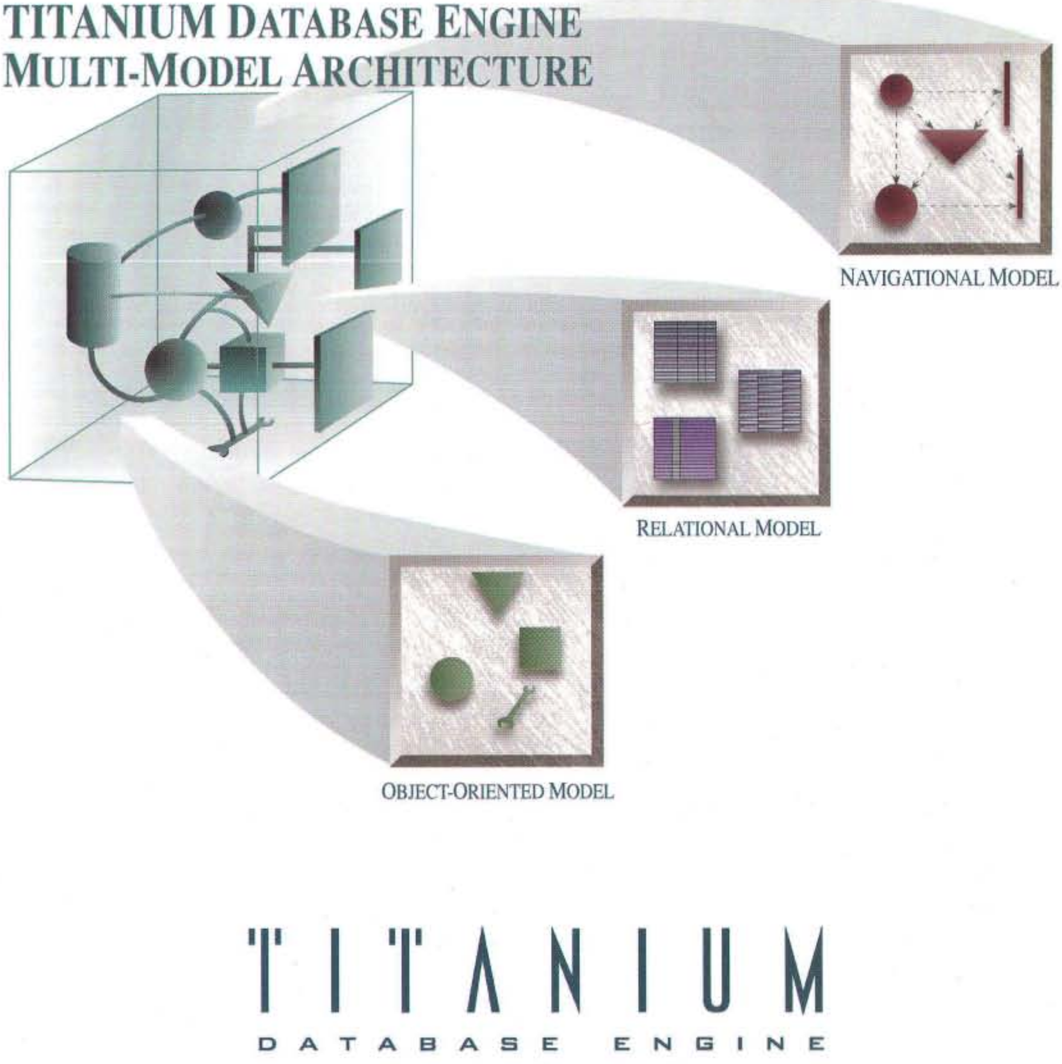
Your Database Shouldn't be Either

Other database engines require design or performance concessions when developing applications with complex data relationships. With the patented TITANIUM® multi-model database engine, you have the freedom to choose the database models which suit your needs, and enjoy unrivaled performance too.

TITANIUM is the only database to allow applications simultaneous access to the database using any of the relational, object-oriented (through C++ Object Classes), and navigational data models. Developers, for example, may use the ODMG-standard object model to manipulate complex objects and relationships, while users and SQL developers view and manipulate the same data as relational tables. Further, each model is pure, avoiding the weaknesses of hybrid models such as object-relational.

TITANIUM's underlying Dynamic Pointer Array (DPA) technology delivers split-second query responses and unrivaled performance. TITANIUM directly supports complex real-world relationships, eliminating the cumbersome intersect tables and redundancy associated with typical databases.

TITANIUM features optimized read/write SQL, BLOBs, nested transactions, as well as compiled triggers and stored procedures. It runs in a megabyte of memory on all PC platforms, with support for a terabyte or more of data.



TITANIUM

DATABASE ENGINE

Optimal Performance for
Complex Development Needs.

AD LINK 146

mdbs®
Micro Data Base Systems, Inc.

For a TITANIUM Technical White Paper, call 800-445-mdbs or 765-463-7200.
E-Mail: info@mdbs.com WWW: http://www.mdbs.com

mdbs and TITANIUM are registered trademarks of Micro Data Base Systems, Inc. ©1998 Micro Data Base Systems, Inc.

(continued from page 30)
date (usually pivot year = "01" is a preferred value); cloning the program that has become Y2K compliant and changing the pivot year value; and processing the standard file through the standard pivot year program; processing the exception file through the cloned program that contains the nonstandard pivot year.

These options are only suggestions. Unique cases will require the application's business and technical experts to devise strategies that solve cases that will not adapt to a 99-year window strategy.

External sorts based on dates require redefining the sort date location and identifying where the year is located. Several sort utility vendors offer add-ons that intercept the sort; append the century per the application rules (the pivot year); sort the file; strip the century; and generate the sort output file in the original format and correctly sequenced based on Y2K criteria.

Testing Date Fields

Window routines must test the date field for all 0s and all 9s. If this is the case, then the expanded field's century value is set to 00 or 99 per the input value (bypassing the pivot date test). Listing Four illustrates this.

When a year is equal to 00, the logic must decide if the value 00 represents an initialization or the year 2000. If a date is defined in the program (YYMMDD, YYDDD, and so on) and the original logic only deals with the year portion of the date field (that is, the YY value), then the unused part of the date field must still be evaluated. If the unused portion of the date (MMDD, DDD) is numeric and not equal to all zeroes or all nines, then the year portion of the date field contains valid data (for this case, year = 00 must be expanded to 2000). If the unused portion of the date is all zeroes (MMDD=0000, DDD=000), then the year equal to 00 is not a valid year and the century must be set to 00 not to 20.

Listing Five illustrates the original code that requires the technique explained above. The revised Y2K-compliant procedure division logic (windowing has been executed elsewhere, and is not shown in this example) is in Listing Six.

When dealing with situations similar to that in Listings Five and Six, it's best to comment out the original logic rather than delete it. The advantages of doing this are:

- To enable the code review to easily identify the obsolete logic;
- To provide a quick reference that aids in resolving problems detected during system testing;
- To permit rebuilding of original logic in a timely manner should the Y2K logic prove invalid or unnecessary.

Compound conditionals can be especially challenging. The original tests should be done before executing the window logic. Unconditional windowing of dates should be avoided. This eliminates the risk of receiving abnormal terminations (ABENDS) due to the failure of considering inclusive/exclusive AND and inclusive/exclusive OR relationships.

Listing Seven illustrates this approach. The year 2000 logic must maintain the original logic's assumption that it is only 100 percent guaranteed that the date values will conform to the date field PIC clause when field B-CLM-STUS-CD = O. When B-CLM-STUS-CD has any other value than O, then the test will not occur and the content in the date fields is unknown and not impacted per the original logic.

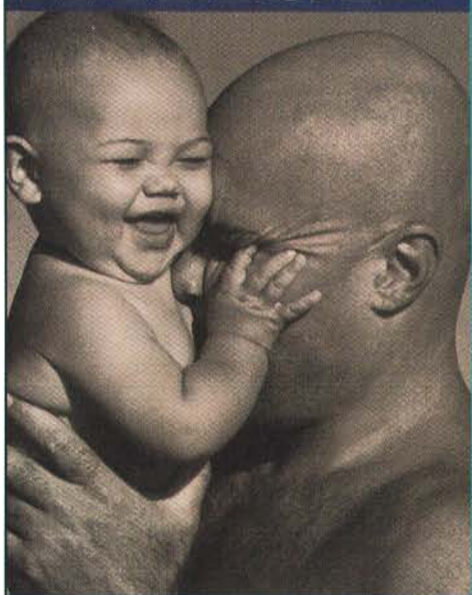
Conclusion

The year 2000 may very well be the largest task an organization ever faces. Selecting the right techniques and tools can significantly affect your time savings. With the windowing method of date conversion, you can achieve compliance in less time with a manageable level of effort. At this point, every hour counts.

DDJ

(Listings begin on page 109.)

HASP® **Protects** Your Software



Your software is your baby – and you want to look after it. You created it, you developed it, you saw it right through to the moment it was ready for market. Now protect it. 50% of business software is stolen; $11 billion of developers' income is lost to piracy.* Is your software a statistic?

All over the world, more developers are protecting against piracy. They're protecting more products, on more platforms, with more security – and selling more as a result. And more of these developers are protecting with HASP.

To see why 25,000 developers worldwide protect with Aladdin, call and order your HASP Developer's Kit now!

HASP – The Professional Software Protection System®

- Rated #1 in industry tests
- Unequalled security
- Unparalleled flexibility
- Genuine ease-of-use and transparency
- Supported under Windows 95, 98 and NT, Mac, OS/2, NEC, UNIX, and LANs
- ISO 9002 quality and reliability
- New - USBHasp!

HASP Protects More

See us at Comdex Spring · April 20-23, Stand 1280

North America 800 223-4277, 212 564-5678, Email: hasp.sales@us.aks.com
Int'l Office +972 3 636-2222, Email: hasp.sales@aks.com
Germany +49 89 89 42 21-0, Email: info@aladdin.de
UK +44 1753 622266, Email: sales@aldn.co.uk
Japan +81 426 60-7191, Email: sales@aladdin.co.jp

Call us for a distributor near you!

* 1997 BSA/SPA Study

USB

1-800-223-4277

www.aks.com

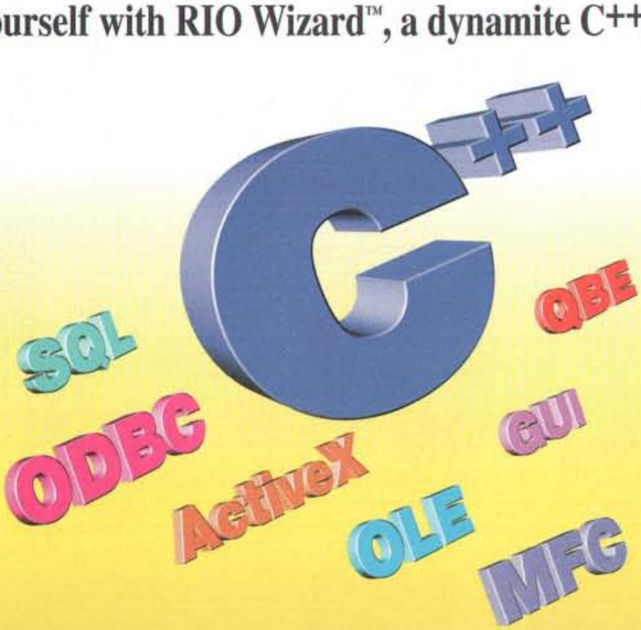
ALADDIN

The Professional's Choice

AD LINK 421

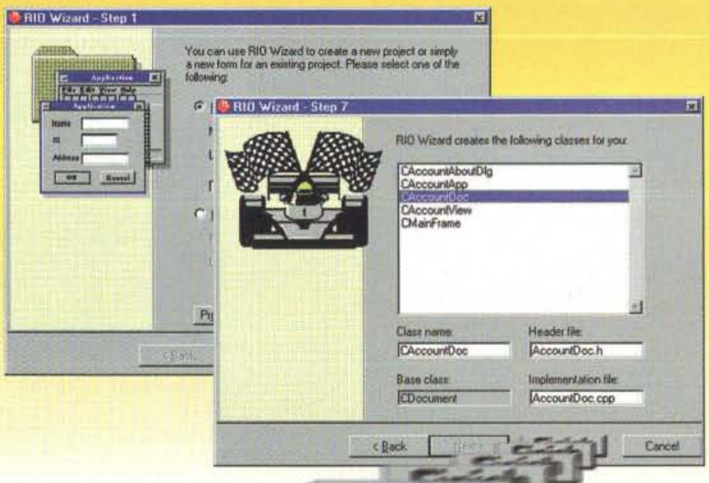
You can Finish your next Visual C++® project ahead of time!

Empower yourself with RIO Wizard™, a dynamite C++/MFC code generator



"RIO Wizard is a useful tool that adds RAD capability to Visual C++. It will save development time, particularly if you write programs that access databases."

– *Eric Sigler*
Object Magazine, March 1998



Delivers the complete running application

- Integrates with Developer Studio
- Has built-in Visual Basic® style GUI builder
- Supports multi-pass visual development of GUI
- Preserves custom code during code re-generation
- Connects application to DAO and ODBC
- Auto-gens database application that supports
  - Single table query
  - Multiple tables Join query
  - Query-By-Example (QBE)
  - Master/details
- Auto-gens data-bound controls (including data grid) in forms
- Supports reuse of GUI design and Visual C++ project
- Lets user define naming rules for projects
- Generates well structured and commented C++/MFC code
- Builds application as exe, dll and ActiveX control
- Royalties free for end-user application distribution (Class library source code sold separately)

FREE evaluation edition available at
http://www.tbridge.com
Try RIO Wizard first-hand and
Experience a new way to work with Visual C++

AD LINK 351

TechBridge Technology Corp.

TechBridge Technology Corp.
275 Renfrew Drive, Suite 103
Markham, Ontario, Canada L3R 0C8

1-800-463-8998
Email: sales@tbridge.com
Phone: (905) 513-7800
Fax: (905) 513-1330

RIO wizard™
Your Best Visual C++ Programming Companion

Designed for Microsoft® Windows NT® Windows® 95

© 1997-1998 TechBridge Technology Corp. TechBridge is a registered trademark, RIO Wizard and the RIO Wizard logo are trademarks of TechBridge Technology Corp. Microsoft, Windows, the Windows logo, Windows NT, Visual Basic and Visual C++ are registered trademarks of Microsoft Corporation. All other names mentioned are used for identification purposes only, and may be registered trademarks/trademarks of their respective owners.

HOW POWERFUL IS THE NEW DELL WORKSTATION?

Viewperf CDRS Benchmark[Ω] Test Workstations – Relative Price Performance[§]		
Dell	WorkStation 400 266MHz w/ELSA Gloria-M Glint MX	$121
IBM®	IntelliStation M Pro Model 26U INGR w/Intense 3D Pro 2200-4T	$178
Compaq	Professional Workstation 6000 Diamond Multimedia FireGL 4000	$242
Sun Ultra	Ultra 1 Creator3D Model 170E w/Creator3D	$446
SGI	O2 R5000/180MHz SCO2	$461
Digital	Personal Workstation 266i w/AccelGraphics AccelPro 2500TX	$424
SGI	Octane 1x175MHz R10000 SI w/Tram	$672

Dell's expertise in industry standard technology gives the Dell WorkStation 400 a price-performance edge over selected models from significant competitors. This chart reflects a 3D-graphics-oriented benchmark modeling an environment similar to mechanical CAD.

§Price per composite score. Lower number indicates greater price value. For more in-depth information, refer to www.specbench.org/gpc/opc/opc.cdrs.summary.html. ΩCopyright 1998, Standard Performance Evaluation Corporation. Information value correct as of print date.

Get ready for takeoff. Because the Dell® WorkStation 400 flies. From its high-bandwidth dual independent bus architecture to its single or dual fast, 266, 300 or 333MHz Intel Pentium® II processors, the word is "go." Even in the midst of the demanding data and graphics applications, it always feels like there's power to spare. And there is, thanks to all the high-performance industry components inside. Like the 3D, financial modeling, CAD, authoring and animation video cards and even enhanced audio capabilities such as integrated 16-bit Sound Blaster Compatible Sound. What's more, the Dell WorkStation 400 has been compatibility tested with leading software applications to help ensure that your critical applications are ready to roll. But the real rush comes from the fact that yours will be custom-built from the ground up. And it's backed by a $9.8 billion global company* that can support it (and your company) in every way. With system engineering, 7x24 technical hardware telephone and online support, consulting, global integration, leasing and asset management. The Dell WorkStation 400 comes with everything you'd expect in a workstation. Everything, that is, except the high-flown price.

ENTRY-LEVEL WORKSTATION

DELL® WORKSTATION 400MT

- 300MHz Pentium® II Processor (Dual Processor Capable)
- 128MB ECC EDO DIMM Memory
- 4.3GB EIDE Hard Drive
- Matrox Millennium II PCI with 8MB Dual Ported WRAM Memory
- 1200HS (17.9" v.i.s.) Monitor
- 24X Max§ Variable EIDE CD-ROM Drive
- Integrated 16-bit Sound Blaster Compatible Sound
- Integrated 3Com® 10/100 EtherLink® Controller
- Factory Installed MS® Windows NT® 4.0 with 1 Year Telephone Support
- 3.5" Floppy Drive
- 3 Year Limited Warranty† with 3 Years Next-Business-Day On-site^ Service

$3855

Order Code: 900038

MID-LEVEL WORKSTATION

DELL WORKSTATION 400MT

- Dual 300MHz Pentium II Processors
- 128MB ECC EDO DIMM Memory
- 4GB Ultra/Wide SCSI-3 Hard Drive (7200 rpm)
- APPIAN Jeronimo J2 Graphics Card
- Two 1000LS (15.9" v.i.s.) Monitors
- 24X Max§ Variable EIDE CD-ROM Drive
- Integrated 16-bit Sound Blaster Compatible Sound
- Integrated 3Com 10/100 EtherLink Controller
- Factory Installed MS Windows NT 4.0 with 1 Year Telephone Support
- 3.5" Floppy Drive
- 3 Year Limited Warranty† with 3 Years Next-Business-Day On-site^ Service

$5854

Order Code: 900013

HIGH-END WORKSTATION

DELL WORKSTATION 400MT

- 333MHz Pentium II Processor (Dual Processor Capable)
- 128MB ECC EDO DIMM Memory
- 9GB Ultra/Wide SCSI-3 Hard Drive
- Advanced 2D/3D ELSA Gloria-XL Graphics Card
- 1600HS Trinitron® (19.8" v.i.s.) Monitor
- 24X Max§ Variable EIDE CD-ROM Drive
- Integrated 16-bit Sound Blaster Compatible Sound
- Integrated 3Com 10/100 EtherLink Controller
- Factory Installed MS Windows NT 4.0 with 1 Year Telephone Support
- 3.5" Floppy Drive
- 3 Year Limited Warranty† with 3 Years Next-Business-Day On-site^ Service

$6449

Order Code: 900039

TO ORDER TOLL-FREE

800-433-3498

TO ORDER ONLINE

www.dell.com

Mon-Fri 7am-9pm CT
Sat 10am-6pm CT
Sun 12pm-5pm CT
In Canada,* call 800-839-0148
GSA Contract #GS-35F-4076D

Keycode #99264

DELL®

www.dell.com

*Prices and specifications valid in the U.S. only and subject to change without notice. †For a complete copy of our limited warranties, please write Dell USA L.P., One Dell Way, Round Rock, TX 78682, Attn: Warranties. §24X Max/12X Min. ^On-site service provided by third-party providers and may not be available in certain remote areas. 3Com and EtherLink are registered trademarks of 3Com Corporation. *Reported revenues for last four fiscal quarters. The Intel Inside logo and Pentium are registered trademarks and MMX is a trademark of Intel Corporation. MS and Windows NT are registered trademarks of Microsoft Corp. Trinitron is a registered trademark of Sony Corporation. Dell and the Dell logo are registered trademarks of the Dell Computer Corporation. ©1998 Dell Computer Corp. All rights reserved.

A Year 2000 Tool Suite

A date scanner and data ager written in Java

Dev Bhattacharyya

Contrary to popular belief, the Year 2000 problem poses a threat not only to mainframe systems, but to desktop and client-server systems. Many of these problems derive from prebuilt routines that are not Year 2000 compliant and are regularly used with the current offering of visual and RAD desktop development tools. Another problem unique to client-server systems is making sure every segment of the application that manipulates data is compliant.

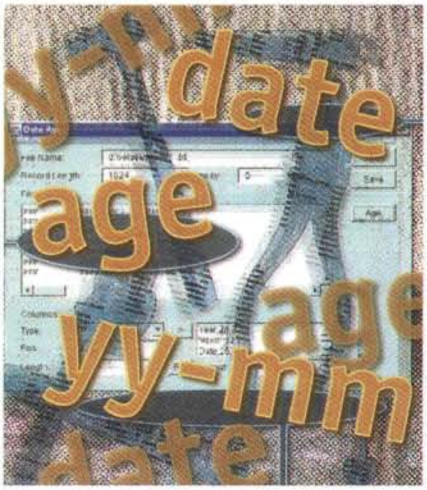
To address these problems, programmers need both tools that scan source code for date-related data structures and functions, and those that create test data from existing production data. In this article, I'll present two such tools. The scanning tool examines an application's source code for date-related areas and offers a preliminary analysis of the target application. The data ager tool lets you manipulate existing production data to create a set of test data. Both the date scanner and data ager (written in Java 1.1) are available electronically; see "Resource Center," page 3.

Date Scanning

The scanner identifies areas in the source code that match a dictionary of date-related words—words often used by programmers to portray dates. The scanner then generates an HTML page containing the source code excerpts with impacted areas hypertext linked to a date-dictionary URL. Not all of the source code in the report are true Year 2000 trouble spots, but it is better to err on the side of caution.

Developers often use real-world terms to describe date variables and functions, so the scanner looks for words like "date," "age," and "yy-mm." Example 1 shows a list of common "real-world" words.

However, as Example 2 illustrates, some developers use words that have no bearing on what they are supposed to represent. Examples of such obscure code are common, and only the developer knows



why he used such deviant names to perform date arithmetic or store dates. To guarantee better analysis, the scanner should include programming language reserved words that are used to declare user defined types and dates—"type," "struct," "class," and so forth.

To avoid identifying nondate trivia as Year 2000 trouble spots (such as "message" which contains "age," and "update" which contains "date"), I include a dictionary of words to ignore. Example 3 contains a list of some nondate-related words that can be potentially misidentified. If the scanned word is in this "ignored words" dictionary, the scanner disregards the word.

Figure 1 presents an abridged description of the *JavaScan* class. The complete Java source code (available electronically) is well documented. Objects derived from standard Java classes in java.io (*File*, *FileInputStream*, *StringBuffer*, and the like) and java.awt (*TextArea*, *Button*, and so on) are used in this class. This applet/application can run on older browsers supporting JDK 1.0.x. To recompile the package under older compilers of the JDK 1.0.x era, you must use the statements that are commented as belonging to JDK 1.0.x.

When the applet/application is loaded, users acknowledge a disclaimer by clicking the Agree button. The text field at the bottom of the applet/application window now has the focus. You must type the valid path and name of the file you want to examine, and click the Open button. The source-code file will be loaded and scanned, and a report is created in the file output.html. You can click on the applet hot-spot to view this page, or you can view the page from your web browser.

The scanning tool scans only a single source file. One possible enhancement to this application would be to scan multiple source files, generate statistics on total lines of code and date-related fields, and output the report to multiple HTML pages.

Aging the Data

There are a number of ways of generating test data, one of which is to "age" current production data. By aging, I mean that all dates are increased or decreased by a specified number of days or years. Each date is increased or decreased by the same amount, maintaining the data's referential integrity between dates. Figure 2 lists the classes used by the prototype for the ager project.

To use the ager, you dump your data containing dates into a text file. The ager tool allows you to specify where the date lies in the text file rows, and the format of the date. The tool also lets you specify the format in which the date will be rewritten (see Figure 3), after which you can reload the data back to where they came from.

I designed this application differently from the date scanner. It is not downward

Dev is a consultant with Visa International's Year 2000 project working on desktop and midrange computers. He can be contacted at devmit97@aol.com.

STOP WASTING TIME DEBUGGING!

Insure++®

Finds Bugs Automatically

Insure++ is a Run-Time Debugger like no other.

It finds the widest variety of programming and memory errors in your programs, automatically!

Insure++ finds all your memory errors, plus uses exclusive Mutation Testing technology to pinpoint algorithmic anomalies and other hard-to-find bugs.

And unlike other tools, *Insure++* gives you the exact location of memory leaks and other errors as they are found in your source code. You'll stop wasting time searching for bugs and start finishing projects faster!

Find Bugs Automatically with *Insure++*

- ✓ Memory Leaks
- ✓ Memory Corruption
- ✓ I/O Errors
- ✓ Logic Errors
- ✓ 3rd Party Errors
- ✓ and more!

AMAZING OFFER!

For a limited time, you can DOWNLOAD Insure++® FREE from our website.

NEW

Mutation Testing Technology Finds the Most Bugs In your C/C++ Code!

Try Insure++ against your current debugging tools. You'll be amazed at how Insure++ finds more bugs automatically, saving you hours--even--days of labor!

For UNIX and Windows NT/95 (Integrates with Visual C++)

ParaSoft® Corporation

AD LINK 294

DOWNLOAD Insure++ today at: www.parasoft.com/insure/eval.htm or call 1-888-305-0041 x 4

AD LINK 424

age	mdy	tomorrow
ccyy	min	year
current	mmm-dd	ymd
date	now	yy-mm
ddmmyy	period	mos
dmy	time	interval
leap	tm	qtr

Example 1: *Common "real-world" words.*

(a)
```
struct dRecord
{
     int xx;
     int yy;
     int zz;
};
```
(b)
```
int dTemp, mTemp, yTemp;
```
(c)
```
Dim dTemp As Variant
```

Example 2: *Words that are not suggestive of what they are supposed to represent.*

breadth	tmp	language
message	update	minim
validate	know	percentage
page	itemindex	manage
bitmap	html	flagerror
femin	idyes	section
image	agent	centimeter

Example 3: *Nondate-related words that can be misidentified.*

(continued from page 36)
compatible to JDK 1.0, and it uses the new features of AWT and the language enhancements of JDK 1.1. I implemented it with Borland's JBuilder 1.0, using Borland's extensions to the AWT classes. These extensions can easily be modified to suit other compilers.

You enter the name of the text data file (complete path information) and click the Open button. Twenty records from the data file (or fewer if the data file is smaller) are loaded as a sample in the *TextArea* component. You then enter the record length to correctly align the 20 records.

You then select the year component and enter the starting point and length in the first row of data in the *TextArea* to mark the year portion of the date. The portions you define are highlighted in the *TextArea*, reconfirming your entries. A button is provided to place the entry in a list for later use. Likewise, you must mark the date, month, and the full date field in the same row.

You then enter the format into the preset date field. The format could be any combination of M, d, y, and separators. Notice the "M" is in uppercase. After all the selections are complete, users click the Age button, and the aged data is saved as "c:\out.txt". A sample 20 rows are displayed back in the *TextArea* for users to confirm whether the results are what they wanted. If the results are erroneous, changing the parameters and rerunning again will achieve the desired output.

This aging tool can alter only one date field at a time. It also can address only one table at a time in a database. The tool is incapable of addressing date-derived fields like invoice numbers or purchase order numbers that may have portions of dates embedded in them.

DDJ

```
Class JavaScan
    Interface:
        Constructor
        Destructor
        GetAppletInformation
        Action
        // Applet Related
        Init
        Start
        Stop
        Run
End Class;
```

Figure 1: *Abridged description of the JavaScan class.*

```
Class DateRoutines
    Interface:
        Constructor
        Destructor
        setDate
        setMonth
        setYear
        setField
        parse
End Class

Class Aging
    Interface:
        Constructor
        Destructor
        getParameter
        // Applet Related
    Action
        Init
        Start
        Stop
        Run
End Class;
```

Figure 2: *Class representations used by the prototype for the ager.*

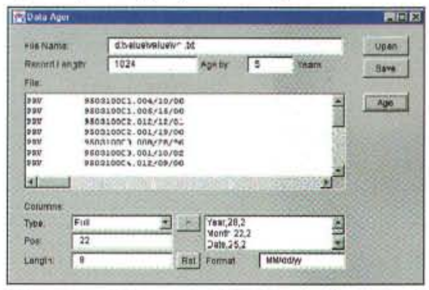
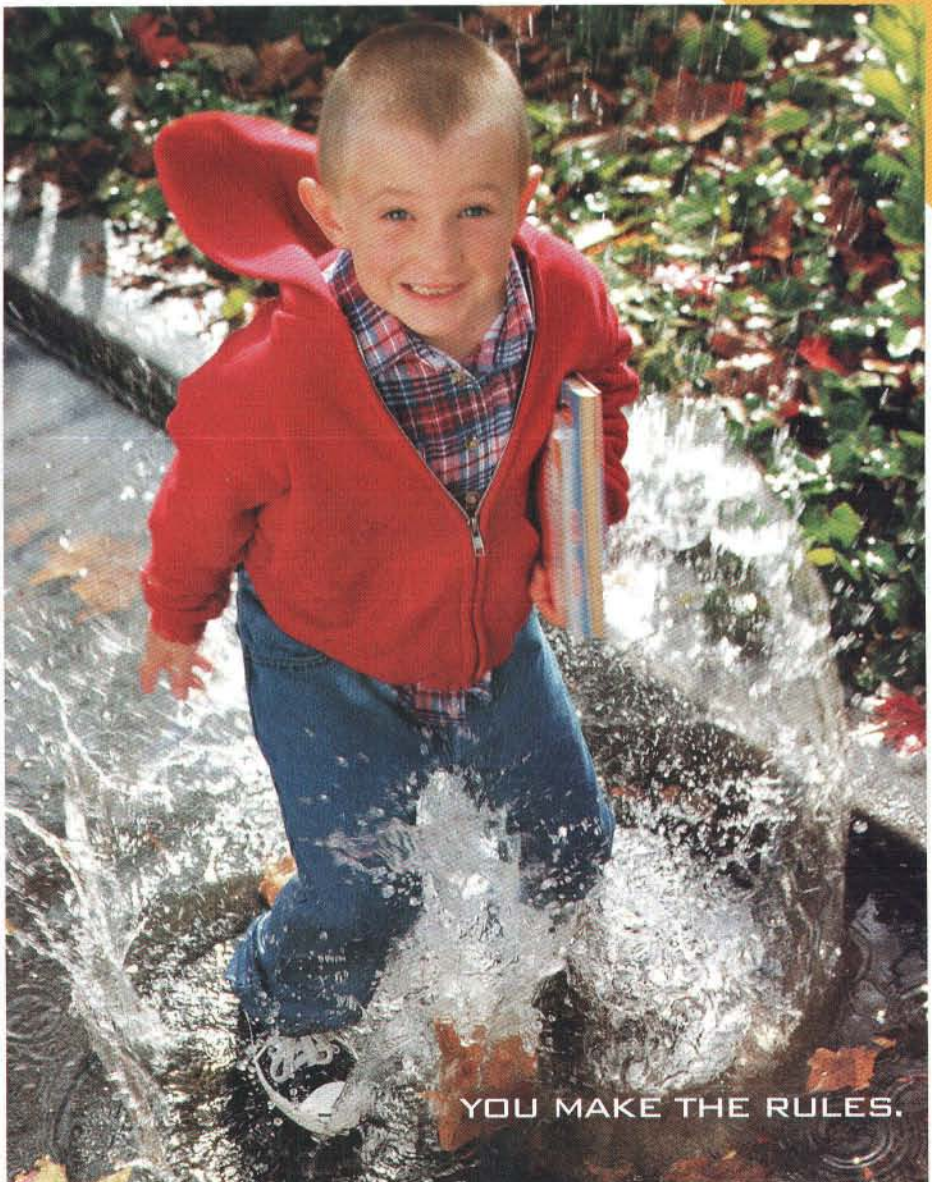


Figure 3: *Running the Data Ager tool.*

"LIVE BY YOUR OWN RULES."



Make your Web site as dynamic as your business and as smart as your customers with ELEMENTS ADVISOR™.
Business rules define your company's policies, practices and ability to compete. Today's dynamic business environment means you often need to jump in and alter the rules that control your Web applications. A rule you can change is the one that says you need to embed your business rules inside application code. Because it separates business rules from application logic, ELEMENTS ADVISOR lets you put new business rules on-line in minutes. And it integrates easily with both new and existing code. So make a splash. Design a self-service application with a new level of interaction. For details visit us on the Web today. See a demo and read *"Business Rules Automation,"* our new white paper on designing "smart" application components at: **www.elements.com**

WE MAKE THE TOOLS.™

ELEMENTS™ FROM NEURON DATA

1·800·876·4900 or 1·650·528·3450 Int'l
www.elements.com

ELEMENTS ADVISOR, ELEMENTS, the ELEMENTS logo and YOU MAKE THE RULES. WE MAKE THE TOOLS. are trademarks of Neuron Data, Inc. All other trademarks are the property of their registered owners. ©1998 Neuron Data, Inc. 027DD

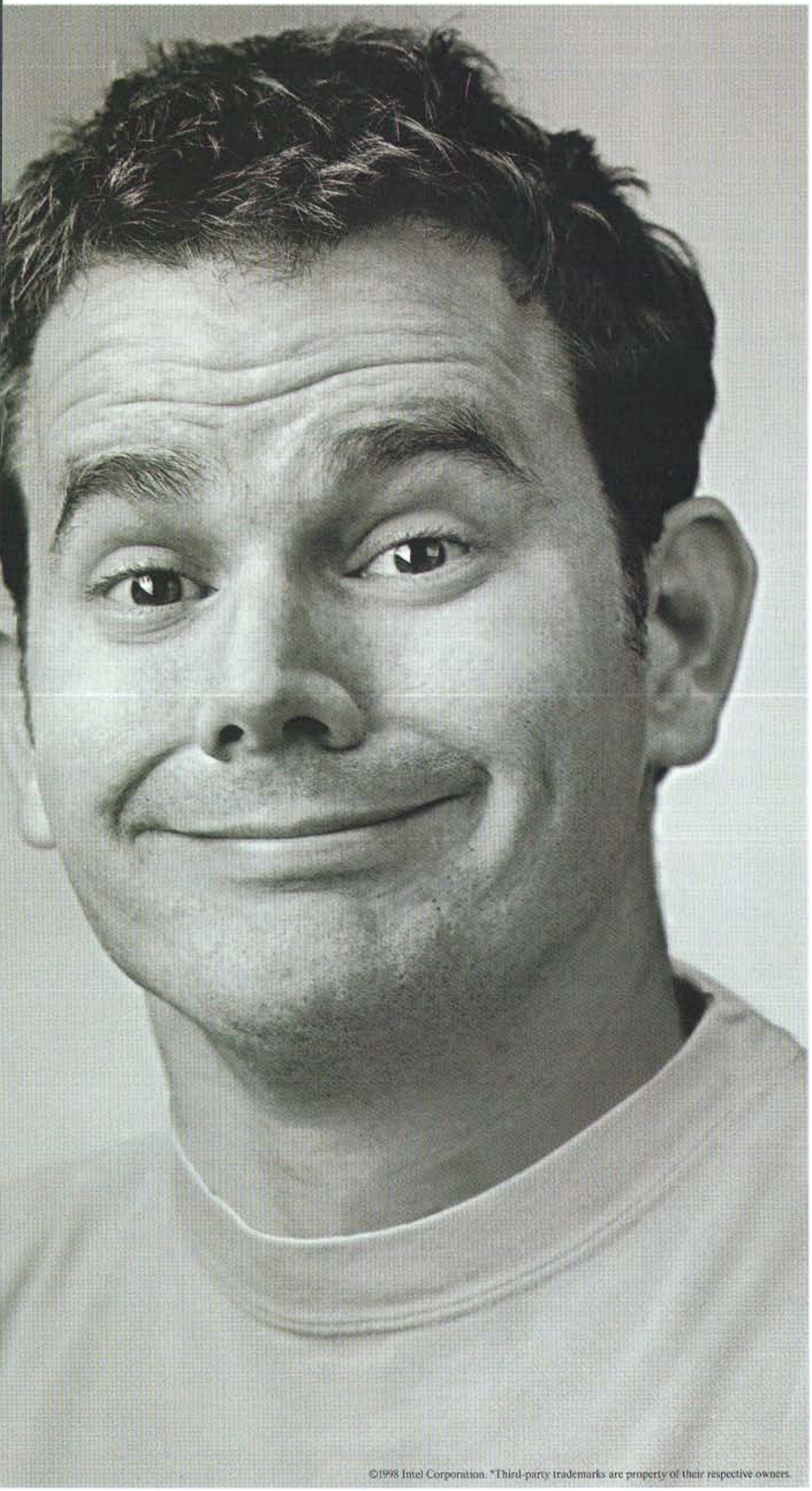
AD LINK 95

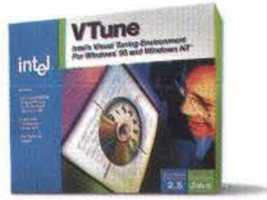
We know *you* don't write slow code so buy it for the genius in the next cube.

```
ocessor on which this program
running.
//////////////////////////
uildProcessorName()

  ProcessorFamilyID=0;
lean CpuidInstructionAvailable;
lean ProcessorIsGenuineIntel;

puidDlg p = new CCpuidDlg(m_pMainWnd);

et Processor Information
First call native method to
execute CPU's CPUID instruction
etCPUID_Data();

cessorFamilyID = ProcessorFamily();
idInstructionAvailable =
CpuidInstructionSupported();
cessorIsGenuineIntel = GenuineIntel();

et processor name
CpuidInstructionAvailable)

' Verify its a recent Intel* CPU
 (ProcessorIsGenuineIntel)

  // 4 is lowest family number
  //  we could see here.
  // All processor signature
  //  info is available
  switch (ProcessorFamilyID) {
    case 0:
    case 1:
    case 2:
    case 3:
      // Programmer's paranoia-
      //  We can't get here!
      p.SetProcName("");
      break;

    case 4:
      // Handle the different types
      //  of Intel486(tm) processors
      //  separately
      p.SetIntelFamily4_ProcName ();
      break;

    case 5:
      // Pentium* processor
      p.SetProcName (ProcessorName
        [PNAM_Pentium_Processor]);

      // Check for MMX(tm)
      //     technology support
      if (ProcessorHasTechnology
        (MMX_TECHNOLOGY))
        p.AppendProcName
          (MMX_Technology_Str);

      // Add in trademark symbols
      p.AppendProcName (
        Trademark_Str);
      p.AppendProcName (
        Registered_Trademark_Str);
      break;

    case 6:
      // Pentium* Pro or
      //  Pentium* II processor
      if (ProcessorHasTechnology
        (MMX_TECHNOLOGY))
      {
        p.SetProcName (
          ProcessorName [
            PNAM_Pentium_II_Processor]);
      }
```

```
    }
    else
    // Processor from the Stone Age,
    //  CPUID inst. not available
    {
      p.SetProcName (ProcessorName
        [PNAM_NO_CPUID]);
    }
}

/////////////////////////////////
//
// Procedure: scale_array_elements
//
// Function:
//  Scale the elements of a floating
//   point array according to the
//   desired scale factor.
//
/////////////////////////////////
void scale_array_elements (float scale_factor)
{
  int  i, j;
  float scale_recip;

// NOTE: VTune showed this method is compiled
//  at execution time by the JIT compiler.

  scale_recip = 1.f / scale_factor;
  for (i = 0; i < size; i++)
    f_array[i] = f_array[i] * scale_recip;

// VTune suggested multiplying by the reciprocal
//  instead of dividing in the loop - multiply
//  is faster.
// Original Code:
//    for (i = 0; i < size; i++)
//       f_array[i] = f_array[i] / scale_factor;
}

/////////////////////////////////
//
// Procedure: GetProcessorCacheDescriptors
//
// Function:
//  Use CPUID instruction to retrieve
//   raw data on the processor's caches,
//   then extract the data into a simple
//   array.
//
/////////////////////////////////
int[] GetProcessorCacheDescriptors
    (int[] Descriptor, int Size)
{
  int   CpuidInfo[][] =
      { {0, 0, 0, 0}, {0, 0, 0, 0} };
```

```
  int   i;

  // Check parameters
  if (Descriptor == null || Size < 1)
    return null;

  // Initialize result array.
  for (i = 0; i < Size; i++)
    Descriptor[i] = 0;

  // Get raw cache descriptors from
  //   native CPUID assembler routine
  p.GetCPUID_CacheData ();
  DescriptorsDefined =
    GetRawCacheDescriptors (CpuidInf

  // Reset descriptor output counter.
  i = 0;

  // Parse the descriptors if available
  if (DescriptorsDefined)
  {
    for (set = 0; set < 2; set++)
    {
      for (dword = 0; dword < 4; dword
      {
        for (abyte = 0; abyte <= 3; aby
        {
          Descriptor[i] =
            CpuidInfo[set][dword]
              & 0xff;
          CpuidInfo[set][dword]
            >>= 8;
          if (Descriptor[i] != 0) i++;

          // Early out on short buffer.
          if (i >= Size)
            return Descriptor;
        }
      }
    }
    return Descriptor;
  }

  // Return null on error exit.
  return null;
}

/////////////////////////////////
//
// Procedure: FpuType
//
// Function:
//  Determine type of floating point
//   unit.
//
// returns:
// FPU_NONE   : if no floating point unit
// FPU_287    : if 287 floating point unit
// FPU_387    : if 387 floating point unit
// FPU_487SX  : only if unambiguously a
//              Intel 487SX NDP
// FPU_486DX_DX2_487SX: if one of In
//              Intel486 DX2, or
//              Intel486 SX processors
//              with 487 SX
// FPU_ON_CHIP : if Genuine Intel proce
//              that supports CPUID
//              and that reports FPU
//              feature flag.
// -1         : if this routine cannot
//              recognize the FPU
//
/////////////////////////////////
int
```





VTune 2.5
Performance Tuning
Environment

Nobody likes to admit it but, the fact is, performance hotspots happen.

And tracing and analyzing them can be tedious. Now Intel's new VTune 2.5 gives you both an inside and system-wide view of how your software executes on Intel microprocessors through a simple, highly graphical interface. Whether you're writing in C/C++, FORTRAN or JAVA*, it shows you the problem areas. VTune even offers tuning advice for improving performance. So there's no more guesswork. See for yourself or, better yet, show it to the genius next door. Download a free VTune evaluation and get the complete technical overview at our Web site.

► developer.intel.com/ad/vt25ad4

©1998 Intel Corporation. *Third-party trademarks are property of their respective owners.

intel®
The Computer Inside.™

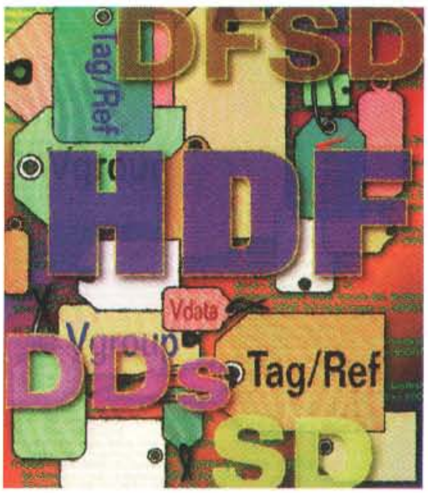
HDF: The Hierarchical Data Format

An all-purpose file format for scientific data

Brand Fortner

Although there is no widespread standard for sharing complex technical and scientific data, the Hierarchical Data Format (HDF), developed at the National Center for Supercomputing Applications (NCSA), is a leading candidate for such a standard. NASA's Mission to Planet Earth Project, an $8 billion effort to monitor the earth's environment, for example, uses HDF as its baseline data standard for its million-gigabyte data archive. The HDF software is available in source and precompiled form for Windows 95/NT, Macintosh, and UNIX, and can be downloaded at no charge from http://hdf.ncsa.uiuc.edu/.

The HDF subroutine library is designed to be easy to use. To write a C character array that represents an eight-bit color image to an HDF file, for example, the only HDF call that's required is *ret=DFR8addimage("myfile.hdf",image1,rows,cols,0);*, which creates and initializes the file myfile.hdf, adds an HDF directory entry to



the file, and stores the image as an HDF object. Although written in C, HDF includes a set of wrapper functions that make it accessible from Fortran.

The NCSA HDF Library

The HDF library has three layers: low-level interface, single-file applications, and multifile applications. Most HDF developers work with the last two layers, which hide the low-level details.

Each layer contains a number of interfaces, each of which contains functions that encapsulate a particular operation or data structure. All functions in a single interface begin with the same letters. Table 1 lists the HDF interfaces, grouped by layer.

Two HDF interfaces support multidimensional arrays, called "scientific datasets" (SDS) in the HDF documentation. HDF 3.3's SDS interface, which has largely replaced the older DFSD interface, supports simultaneous access to more than one file at a time.

The SDS interface supports multidimensional arrays containing 8-, 16-, or 32-bit signed or unsigned integers, or 32- or 64-bit floating-point numbers. This interface lets you specify the coordinate system used by the data, and the scale and units associated with each axis. You can specify how values should be formatted for display, a label for each variable, and store the maximum/minimum values for reference.

The Vdata model (VS, VSQ, and VF interfaces) makes it easy to store tables of data in HDF files. Each table consists of a series of records, each of which contains a series of fields. Each field can support its own number type. Valid field types include 8-, 16-, 32-bit signed and unsigned integers, 32- and 64-bit floating-point numbers, and ASCII characters.

Vdata tables use three kinds of identifying information:

Brand is the chairman of Fortner Software LLC and coauthor of Number by Colors *(with Ted Meyer) and* The Data Handbook, *both from Springer-Verlag. He can be contacted at brand@fortner.com.*

Make Sense of Complex Data

Graph Layout Toolkit

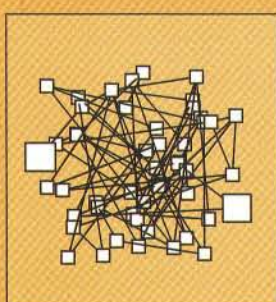
Now available for Java!

The Graph Layout Toolkit's powerful component technology gives you the tools to visualize complex relational information. With sophisticated layout technology, the positions of objects and their connections in diagrams, networks and schemata are automatically calculated in seconds. The Graph Layout Toolkit's unparalleled techniques are a result of Tom Sawyer Software's pioneering development of graph layout components.

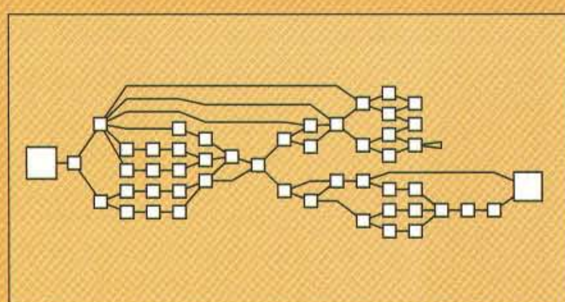
Hundreds of tailorable features are encompassed in the Graph Layout Toolkit's four layout libraries: Circular, Hierarchical, Orthogonal and Symmetric. These libraries provide the visual enhancement required to make your application more effective.

Benefits of the Graph Layout Toolkit

- Automates the positioning of complex data eliminating manual layout.
- Scales automatically to support an unlimited number of nodes allowing complete access to complex diagrams.
- Graph editing operations such as cut, copy, and paste provide smooth manipulation of graph components.
- Automatic edge label placement allows immediate interpretation of the drawings.
- Ability to expand and collapse sub-drawings provides easy navigation.



before



after

download a demo at

www.tomsawyer.com

AD LINK 301

Tel: +1 510 848 0853 E-mail: info@tomsawyer.com

Tom Sawyer SOFTWARE

(continued from page 42)

- A name describes the origin and contents of a table.
- A class identifies the meaning of data.
- Field names identify the individual fields that make up a record.

Example 1 is a Vdata table with three fields and four records.

The Vdata VS interface provides routines for reading/writing tables, the VSQ interface routines for querying *Vdatas*, and the VF interface routines for manipulating individual Vdata fields.

The Vgroup interface provides routines for reading/writing groupings of HDF data objects in a particular HDF file. Each Vgroup may contain any number of other HDF data objects—even other Vgroups. In addition to its members, a Vgroup may also be given a Vgroup name and Vgroup class. Every function of Vgroup begins with the prefix V. The VH interface contains routines for simple creation of Vgroups and Vdatas.

HDF Examples

Listing One (listings begin on page 109) takes an eight-bit raster image (stored as a two-dimensional array of eight-bit values), creates a linear grayscale palette, and stores this to an HDF file.

Scientific data is often stored in multidimensional arrays. For example, a biologist might measure the plant density in each square of a large grid, or an astronomer might have a computer simulation that provides the density at each point in a large gas cloud.

Listing Two illustrates how to store a three-dimensional array of data (including some related comments) in an HDF file using the SD interface. Listing Three retrieves that same data. Listing Three doesn't need to know in advance the size or dimensionality of the data—it can read all of that information from the file.

The HDF Disk Format

An HDF data file consists of a magic number (the control characters ^N^C^S^A), a directory that points to data elements, and the data elements themselves.

The HDF directory entries—known as "Data Descriptors" (DD)—are organized into a series of DD blocks. The HDF directory (or DD list) consists of a series of these blocks. Each DD block contains two bytes indicating the number of DDs in that block, a four-byte integer with the location of the next DD block (or 0 if this is the last one), and then the actual directory entries. Figure 1 shows the beginning of a simple HDF file with one DD block containing two DDs.

The location of the next DD block entry (like all locations within an HDF file) is given in bytes from the beginning of the HDF file. Since all locations are given

Interface	Description	Example Routines
(a)		
H	low-level I/O, directory, query	Hopen, Hread, Hwrite, Hcreate
HDF	new version of low-level routines	HDFopen, HDFclose
HE	low-level error reporting	HEreport, HEprint
(b)		
DFR8	read, write 8-bit raster images	DFR8addimage, DFR8getdims
DFP	read, write color palettes	DFPaddpal, DFPgetpal
DF24	read, write 24-bit raster images	DF24addimage, DF24setdims
DFSD	single file scientific dataset	DFSDputdata, DFSDsetdimscale
DFAN	text annotation records	DFANputlabel, DFANgetdesc
(c)		
SD	multifile scientific dataset	SDstart, SDcreate, SDdiminfo
NC	netCDF interface	nccreate, ncopen, ncvardef
VS	Vdata interface	VSattach, VSfdefine, VSgetid
VSQ	Vdata query	VSQuerycount, VSQueryname
VF	Vdata fields inquiry	VFfieldsize, VFfieldname
V	Access, Specify, Inquire Vgroups	Vattach, Vstart, Vsetname, Vgetid
VH	Simple Vdata, Vgroup creation	VHmakegroup, VHstoredata

Table 1: *HDF interfaces by layer: (a) low-level layer interfaces; (b) single-file application layer interfaces; (c) multifile application layer interfaces.*

Tag Name	Tag#	Comments
(a)		
DFTAG_RLE	011	Specifies the run length encoding used for image.
DFTAG_TID	102	Tag identifier: text string of user defined tag.
DFTAG_DIL	104	Data identifier label: used for titles of elements.
DFTAG_DIA	105	Data ID annotation: lengthy text annotation block.
DFTAG_NT	106	Number type: values are float, integer, text, etc.
DFTAG_MT	107	Machine type: specifies IEEE or local computer type.
(b)		
DFTAG_ID	300	Image dimension: gives X and Y size of assoc. image.
DFTAG_LUT	301	Lookup table: color lookup table for assoc. image.
DFTAG_RI	302	Raster image: points to actual image data.
DFTAG_RIG	306	Raster image group: lists all DDs assoc. with image.
DFTAG_LD	307	LUT dimension: size of color lookup table.
DFTAG_CFM	311	Color format: grayscale, pseudocolor, RGB, HSI, etc.
(c)		
DFTAG_SDD	701	SDS dimension: dimension sizes of scientific dataset.
DFTAG_SD	702	Scientific data: points to actual scientific dataset.
DFTAG_SDS	703	Scientific data scales: Arrays for X,Y,Z locations.
DFTAG_SDL	704	SD labels: text describing data and dimensions.
DFTAG_SDU	705	SD units: text with units for data and dimensions.
DFTAG_SDF	706	SD format: text with format code for displaying data.
DFTAG_SDM	707	SD max/min: minimum/maximum valid values for data.
DFTAG_SDC	708	SD coordinate system: text string defining CS.
DFTAG_NDG	720	Numeric data group: lists all DDs assoc. with SD.
(d)		
DFTAG_VG	1965	Vgroup: provides general-purpose grouping of DDs.
DFTAG_VH	1962	Defines the structure of a Vdata data element.
DFTAG_VS	1963	Points to Vdata data element using DFTAG_VH format.

Table 2: *A selection of NCSA defined HDF tag types: (a) utility tags; (b) raster image tags; (c) scientific dataset tags; (d) Vgroup/Vdata tags.*

Location	Value	Comment
0 to 3	^N^C^S^A	Unique HDF Number (^N = control-N, etc.)
4 to 5	2	Number of DDs in DD block
6 to 9	0000	Location of next DD block (none here)
10 to 21	—	Data Descriptor #1
22 to 33	—	Data Descriptor #2

Figure 1: *Beginning of a simple HDF file.*

```
1  2  3  4  5  6  7  8  9  10  11  12
–Tag–I–Ref–I——Offset——I——Length——I
```

Figure 2: *HDF Data Descriptor layout.*

in 32-bit signed integers, the maximum size of a self-contained HDF file or of a single data element is about two gigabytes.

Every data element (image, array, annotation, and so on) in the HDF file has an associated Data Descriptor in the DD list. Every DD is 12 bytes long and has four fields; see Figure 2.

The *Tag* field, which defines the data element type, is defined as a 16-bit unsigned integer, which means there are 65,535 possible types of data elements. (0 is not a legal tag number.) The possible tag values are divided into three ranges: Those below 32,768 are assigned by NCSA, those above 64,999 are reserved, and the rest can be user defined. Table 2 shows some of the data element tag types.

The HDF library assigns a reference number (*Ref*), which identifies data elements with the same *Tag*, to each data object as it is written to the file. The library keeps track of the reference numbers that have been used in the file and guarantees that no two data objects will have the same tag and reference number in the same file. A *Tag/Ref* pair can then be used as a unique ID (known in HDF terms as a "data identifier") to unambiguously identify particular data elements.

HDF is both a subroutine library and a file format

It is allowable for two objects to carry the same tag (such as our two raster images), or for two objects with different tags to carry the same reference number (such as a raster image and an SDS). However, you shouldn't assume that objects with the same reference number have any special relationship. Instead, HDF uses Groups to define relationships.

Relationships between objects are handled with one of three group objects. The Raster Image Group Tag (DFTAG_RIG, tag 306) groups objects associated with a particular raster image. The Numeric Data Group Tag (DFTAG_NDG, tag 720) groups data objects associated with a particular scientific dataset. The Vgroup tag (DFTAG_VG, tag 1965) supports a generalized grouping of all types of data objects, even other Vgroups.

Group objects contain a list of *Tag/Ref* pairs for each object in that group. From that, a program can locate the DD entry for the object, which gives the offset and size of the object in the file. (An *Offset* field gives the location of the data element in bytes from the start of the file, while the *Length* field gives the length of the data element in bytes.)

The Future of HDF

Mike Folk

Scientists typically use one computer to render results for visualization, and another to further analyze and visualize the data. Furthermore, they frequently share data with colleagues. The need to use a mix of computers and transport large amounts of data among many different computers was an early data management problem for many scientists at the University of Illinois National Center for Supercomputing Applications (NCSA).

In response, NCSA developed the Hierarchical Data Format (HDF) in 1988. NCSA HDF is a portable, self-describing data format for moving and sharing scientific data in networked, heterogeneous computing environments. HDF can store several different kinds of data objects, including multidimensional arrays, raster images, color palettes, and tables. It allows individual scientists to mix and group different kinds of data in one file, according to their needs. NCSA provides a library of APIs for reading and writing HDF as well as workstation tools for visualizing data stored in HDF files.

Although HDF has evolved to meet new requirements, support new kinds of scientific data and applications, and operate effectively in new computing environments, some important new requirements seriously test the original design of HDF. Examples of these new requirements include:

- The need to store very large objects (the current HDF limit is two gigabytes).
- The need to store large numbers of objects (the current limit is 20,000 objects).
- More general, flexible data models.
- Performance improvements.
- Compatibility with object-oriented databases and distributed-object technologies.

To address these new needs, the NCSA HDF project is working on a prototype for the next generation of HDF, code-named "HDF 5." Current plans call for three fundamental changes in HDF 5:

Unified data model. The proposed data model will support only one datatype: a multidimensional array of atomic elements. The new object will have two required attributes: dimensionality (the number and sizes of dimensions) and a data type (a definition of the array elements type). More data types will be supported, including record structures. Objects will include optional user-defined attributes of the form "parameter = value." Users will specify optional physical storage schemes for the data, such as compressed storage and possibly an indexed structure. For backward compatibility, the new HDF object is designed so that all current objects can be defined as subtypes of this basic object type.

New file structure. The new file structure will support files and objects of any size and any number of objects. The internal structure for describing objects is simpler than the current structure and should provide faster, easier access to objects.

New I/O library. In planning the next-generation HDF library, NCSA developers hope to exploit similarities between HDF and other popular scientific data formats by building a system that understands a variety of different data models and formats. APIs at the top level allow programs to view data according to a variety of different data models. These APIs communicate with the middle layer that interprets their requests in terms of a common model.

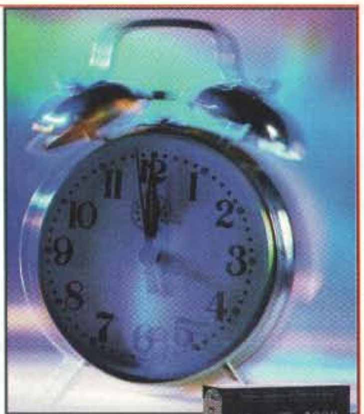
The service layer consists of different file-format drivers, each of which reads from or writes to one file format. Each driver has a well-documented interface for transferring objects and lists of objects to the higher arbitration layer. Possible drivers in the first implementation include HDF, BigHDF, netCDF, and FITS.

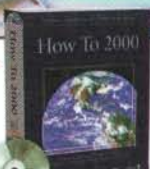
DDJ

Mike is the HDF Project Manager at NCSA. He can be reached at mfolk@ncsa.uiuc.edu.

January 1, 2000

The Alarm is Set... Are You?





How To 2000™
Professional Resource Guide

$49.99

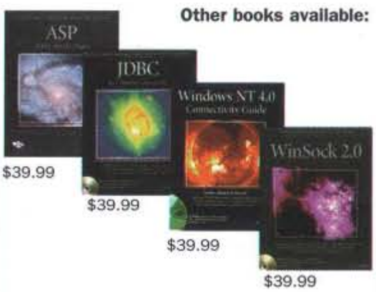
by Raytheon,
the Global Technology Leader.
Experts trusted by the U.S. Government and companies of all sizes!

Wake Up—Address Your Year 2000 Compliance Problem Today!

Visit your local book retailer
or call to order:
(800) 762-2974

To view a sample chapter
visit us at
www.idgbooks.com

Other books available:



$39.99
$39.99
$39.99
$39.99

Raytheon

How To 2000 is a trademark of Raytheon E-Systems, Inc. The IDG Books Worldwide logo is a trademark under exclusive license to IDG Books Worldwide, Inc., from International Data Group, Inc. All other trademarks are the property of their respective owners.

IDG BOOKS WORLDWIDE

AD LINK 127

Both Raster Image Groups (RIG) and Numeric Data Groups (NDG) consist of nothing but a list of data identifiers, each four bytes long. Each is restricted to contain only certain types of tags.

Vgroups do not have this limitation: They can contain any collection of Data Identifiers, including a Vgroup Data Identifier. This lets you construct tree-structured organizations similar to disk directories.

Example 2 shows an HDF file with Vgroups. (I have cheated a bit in this table, because the structure of Vgroup data elements is a bit more complicated than what is shown.)

This file has three objects, all listed in the DD list. The first VGroup refers to the second VGroup, which refers to the image size object. As with pointers in C, it's possible to create circular links; it is the responsibility of the data producer to ensure that the structure makes sense for the application.

HDF Scientific Datasets

The most important data objects in HDF files are scientific datasets, which are multidimensional arrays of numbers. The array can have any dimensionality up to 32,767, and consist of floating-point or integer values.

The DFSD Interface. An SDS, whether created by the DFSD or SD interface, consists of several data objects that are associated with the SDS. These data objects include attributes, numerical scales, the actual data, and so on. In the DFSD interface, all the DDs of the data objects that are associated with a particular SDS are listed in a numeric data group (DFTAG_NDG). Some older HDF files use the roughly equivalent scientific data group (DFTAG_SDG) to collect its members. In either case, only scientific data set tags and a few utility tags are allowed in a numeric data group.

The data identifiers stored in the NDG for this SDS are in Example 3, along with

Vdata Name:	Flux Table		
Vdata Class:	Class_Table		
	Field #1	Field #2	Field #3
Field Types	int16	float32	char[5]
Field Names	ID#	Flux	Name
Record #1	1	2.34	CygX1
Record #2	2	-89.43	HerX1
Record #3	3	0.0023	CygX3
Record #4	4	1.115	Vela

Example 1: *Example Vdata table.*

Location	Length	Value	Comment			
0 to 3	4	^N^C^S^A	Unique HDF number			
4 to 5	2	3	Number of DDs in block			
6 to 9	4	0000	Loc. of next DD block			
		Tag	Ref	Offset	Length	
10 to 21	12	300	001	46	4	DD #1 (Image size)
22 to 33	12	1965	001	50	22	DD #2 (Vgroup #1)
34 to 45	12	1965	002	72	22	DD #3 (Vgroup #2)
		X Size	Y Size			
46 to 49	4	300	200	Image size		
		Name	Tag/Ref			
50 to 71	22	'Vone'	1965/002	First VGroup		
72 to 93	22	'Vtwo'	300/001	Second VGroup		

Example 2: *Organization of an HDF file with two Vgroups.*

Tag Name	Value(s)			Comments
DFTAG_DIL	'Neutron Star Accretion Simulation'			Title
DFTAG_SDD	2 Dimensions: 4 10			Number and size of dimensions
	Float32	Float32	Float32	Number types
DFTAG_SDS		-1.67...1.70	30.75...32.11	Numerical scales
DFTAG_SDL	'Density'	'Time'	'Radius'	Labels
DFTAG_SDU	'gm/cm^3'	'sec'	'km'	Units
DFTAG_SDF	F7.4	F5.2	F5.2	Numerical Formats
DFTAG_SD	'0.5872,0.5872, ... ,1.4319,1.4352'			Actual data

Example 3: *Data objects for an example SDS.*

the information pointed to by the data identifiers. I ignore in this list the actual format of the data identifiers, and concentrate instead on the information that they point to. The *Ref*, *Offset*, and *Length* fields for each DD are not shown.

The DFTAG_SDD tag says that the data set consists of two dimensions, sized four and ten elements, respectively, where the data and both scales consist of floating-point numbers. The numerical scales are set by DFTAG_SDS, with a four-element array for the first dimension and a ten-element array for the second dimension.

The SD Interface. The basic ideas behind the SD version of the SDS are the same as those in the DFSD SDS, but some changes were needed in order to achieve the major design goal: compatibility with netCDF. This driver gave rise to three distinct differences between the two implementations.

First, rather than using a specialized tag such as DFTAG_NDG to group its elements, the SD interface uses a general-purpose Vgroup. Using a Vgroup allows for more flexibility in the organization of the SDS without a complete redesign. Second, the handling of dimensions has changed considerably, in that they can now have their own attributes. Third, the SD interface allows the use of user-defined attributes, not just the ones that are predefined. What follows is a list of the different styles of SDS, in order of their adoption into the HDF library.

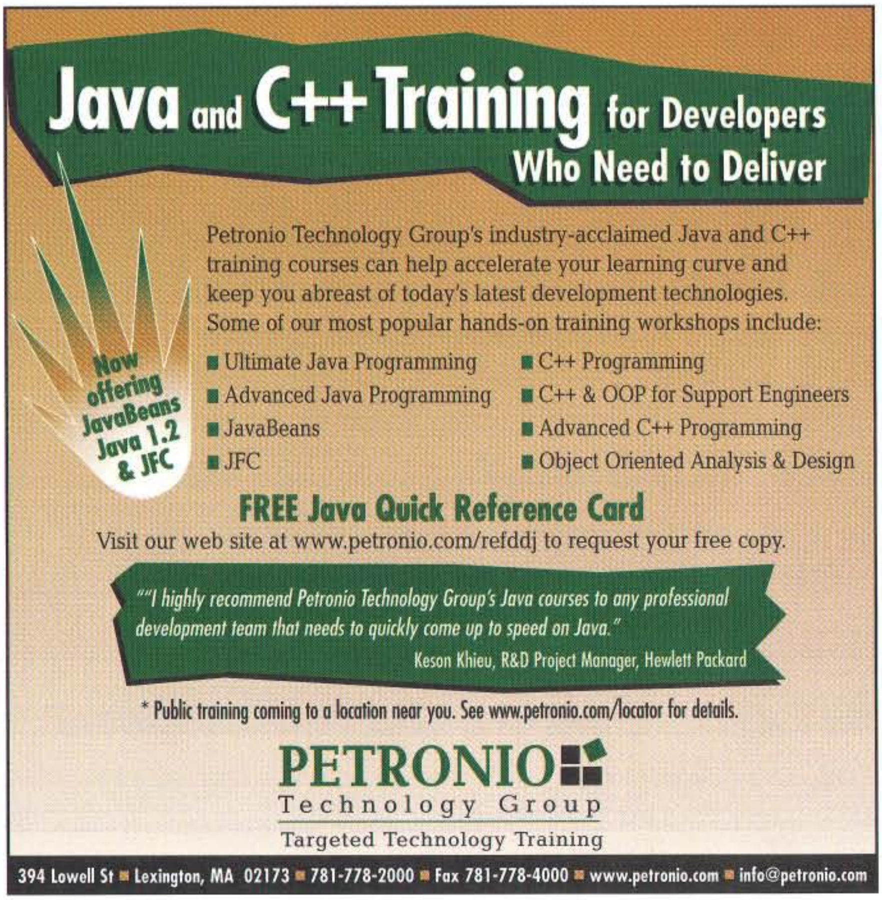
- Scientific Data Groups (SDG), an obsolete form of SDS that only supports 32-bit floating-point arrays.
- Numeric Data Groups (NDG), the disk format for NDG groups (tag value = DFTAG_NDG).
- Multifile SDS, the SD interface (after the prefix of the subroutine calls) uses Vgroups, Vdatas, and many of the parts of the NDG to create structures that are compatible with netCDF data structures.

These subroutines allow several HDF files to remain open simultaneously, something not possible with the earlier libraries. This interface can also assign arbitrary attributes to SDS. In the documentation they are also referred to as "New Style" SDS, or (confusingly) just SDS. The multifile interface will recognize all older forms of the SDS.

HDF Vdata

Vdata tables requires the use of two tags: DFTAG_VH for naming the columns of values; and DFTAG_VS for pointing to the actual Vdata data itself. There is no explicit grouping needed for Vdata; related DFTAG_VS and DFTAG_VH records have the same reference number. This is the only case where the HDF libraries use the

AD LINK 408



AD LINK 400

reference numbers explicitly to associate two data elements.

Example 4 shows the Vdata tags needed for Example 1. Again, we focus on the information pointed to by the tags and ignore the actual binary formats. In particular, we have assigned names to the various fields defined in the data record pointed to by DFTAG_VH. These field names are also used in the HDF documentation.

Tag Name	Field Name	Value(s)			Comments
DFTAG_VH	<nvert>		4		Number of records
	<isize>		11		Row width in bytes
	<nfields>		3		Number of fields
	<type>	int16	float32	char[5]	Field number types
	<isize>	2	4	5	Field size in bytes
	<offset>	Ø	2	6	Byte offset of field
	<fldnmlen>	5	4	4	Length of field Name
	<fldnm>	'ID.No'	'Flux'	'Name'	Field names
	<namelen>		19		Length of Vdata name
	<name>	'Vdata Example Table'			Vdata name
DFTAG_VS		1,2.34,'CygX1',....,'Vela'			Actual data

***Example 4:** Vdata tags for Example 1.*

The DFTAG_VH data element defines a Vdata with three fields (columns) and four Vdata records (rows). The values in these fields are defined as short (two-byte) integer, four-byte floating point, and five-character ASCII text, respectively. In addition, each field has an ASCII text name. Note how the entire Vdata definition can be named (<name>). This is unusual; most HDF data objects require a separate tag (DFTAG_DIL) to get a name.

The data pointed to by the DFTAG_VS tag is stored packed. In this example, each record requires 2+4+5=11 bytes, and the complete table takes 11×4=44 bytes.

HDF Extended Tags

The organizational features of HDF are so powerful that it is usually possible to store a complete data product granule in a single HDF file. There are, however, a couple of problems with very large single-file data products.

The first problem is that HDF files are limited to two gigabytes in size. The second problem is that HDF data elements were not initially designed to be appendable. Normally, data elements are required to be in contiguous storage and they are packed right next to each other. Therefore, to make a data element bigger, it needs to be copied to the end of the file where there is room to grow, leaving a gap in the file.

To get around these limitations, NCSA has defined "extended tags," which allow data element to be spread among multiple locations in the HDF file or even in a completely separate file. A data element corresponding to any NCSA defined tag value can (in principle) be converted to an extended tag, although this functionality is currently only supplied for SDS and Vdata.

An extended tag DD does not point directly to the data, as do normal tags. It instead points to a data object defining where the data is and how it is stored. This data object may point to the beginning of a linked list of data blocks that contain the entire data record. This way, a data record can be lengthened just by adding an additional data block; the entire HDF file does not need to be rewritten.

Alternatively, the extended tag record could define the data element as being stored in an external element in another disk file. This feature (also found in CDF), lets users get around the two-gigabyte total file size limitation. The reference to the external file is explicit: That file must exist exactly as specified by directory and name. This is not compatible with moving files extensively between disparate computers.

DDJ

(Listings begin on page 109.)

ICE Browser is the COOLEST Java software component for rendering HTML. ICE Browser complies with the Java Beans standard and can be used as a simple standalone web browser, embedded in other Java applications and extended for use in custom built web browsers.
ICE Browser is lightweight (only 120kb), fast and robust.

ICE Browser is used by OEMs to build custom browsers, help systems and web-centric applications.

Binary and source code licensing is available.

Try it for *FREE* for 60 days from http://www.icesoft.no/

ICEsoft AS
High Technology Centre
N-5008 Bergen
Norway

icesoft@icesoft.no
www.icesoft.no

Features:

- HTML 3.2 compliant.
- Tables.
- Frames.
- HTTP Authentication.
- Applets.
- Forms.
- Forgiving HTML Parser.
- Printer support.
- Java archive (JAR) format support.
- Superb scrolling performance.
- Client slide image maps.
- Multithreaded for asynchronous loading and parsing of documents.
- Asynchronous image and applet loading.
- Persistence – serializes current state.
- Background images.
- Document history.
- Lightweight (only 120kb).
- Fast rendering!

Microsoft® Windows® CE 2.0 is
the greatest embedded operating
system in the world! You'll get to leverage
the billions of benefits supplied by a supported,
standardized embedded platform.

Marketing Truth

Working with the componentized operating
system is a snap with the award-winning
visual development environment!
Wow! Talk about a giant
leap for mankind!

Windows CE 2.0
is a new operating
system for my
embedded applications.
I can select only the parts
I need. Lots of hardware and
software companies support it.

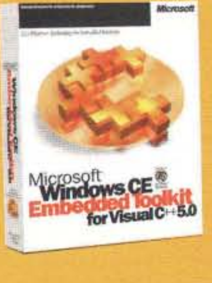
Developer Truth

The Microsoft Windows CE Embedded Toolkit for
Visual C++® 5.0 supports the Win32® API and includes
an integrated development environment.

This is good.

Microsoft® Windows® CE:

The support you need to succeed.



Microsoft®

The Super-Fast, Extremely Vast Personal Storage Drive.™

$299 external drive

1 gig Jaz® cartridges for as low as $89.95**

Each cartridge has a huge one gigabyte capacity

1-Step™ backup software protects as much as 2 gigs (compressed†)

Access time

15.5ms read/17.5ms write

Average seek time

10ms read/12ms write

Maximum sustained transfer rate

6.62MB/sec. max

† Assuming 2:1 compression ratio. Actual compression will vary with file and hardware configuration.

†† Performance will vary when using 1GB cartridges.

©1998 Iomega Corporation. Iomega, the Iomega logo, and Jaz are registered trademarks, and "The Super-Fast, Extremely Vast Personal Storage Drive," "Because It's Your Stuff," and 1-Step are trademarks of Iomega Corporation. *Kai's Photo Soap and Kai's Power GOO are trademarks of MetaCreations, Inc. All other trademarks are the property of their respective holders. The views expressed herein are the views of the endorser and are not the views of, and do not constitute an endorsement by, any person or firm for whom the endorser has provided services. **When purchased in multi-packs. Prices listed are estimated street prices. Actual prices may vary. 2GB capacity where 1GB=1 billion bytes. The capacity reported by your operating system may differ, depending on the operating system reporting utility.

"How we fit 728,000 fractals, 6 cabinets of Soap,* and a whole mess of GOO* into a 4 INCH SQUARE."

BEN WEISS

Imaging Scientist

MetaCreations

They call Ben Weiss an "Imaging Scientist." Sounds cool, but he's much more than that. He's a top U.S. scholar, a pianist, a surfer, and one of MetaCreations' key programmers behind Kai's Photo Soap™ and the world's #1 graphics title, Kai's Power GOO.™ And whether Ben's backing up gigantic files, transporting or mirroring his work, he needs space. Big space. Space like the high-capacity Jaz® drive.

So while Ben's work is big enough to show up on computers around the world, thanks to handy, removable Jaz disks, it's small enough to take with him to the beach.

Check out more of Ben's story at www.4inchsquare.com/ddj.

The high-performance standard in removable storage with one million Jaz 1GB drives already out there.

jaz 2GB

Two times the capacity and over 30% faster than the 1 gig Jaz® drive.

$649 external drive

2 gig Jaz cartridges for as low as $149.95**

Each cartridge has

two monster gigabytes of capacity

2GB is read/write compatible with 1GB††

1-Step™ backup software can protect

up to 4 gigs (compressed†)

Access time

15.5ms read/17.5ms write

Average seek time

10ms read/12ms write

Maximum sustained transfer rate

8.7MB/sec. max

For information about connecting

your Jaz drive to a Mac or PC,

see your reseller or visit us at:

www.iomega.com

BECAUSE IT'S YOUR STUFF.™

iomega

AD LINK 394

Algorithms for High-Precision Finite Differences

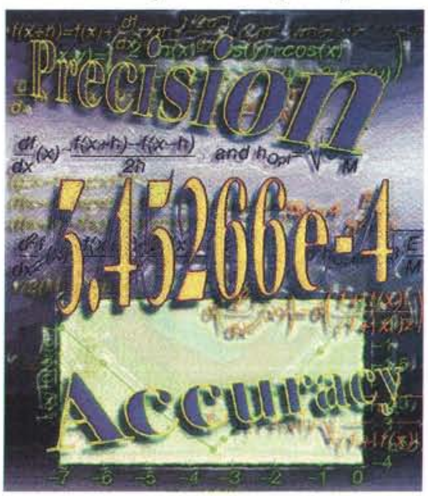
Improving the accuracy of numerical methods

Michael Herstine

In developing applications for science and engineering, code reuse is just as desirable as in other areas of software development, but can often have obstacles peculiar to applied mathematics. In particular, it is common to use the results of legacy code as input to new, higher-level algorithms. In practice, this amounts to coding numerical methods where a single function evaluation (from the standpoint of the algorithm) is, in fact, a run of large amounts of code. Often, that code is single-precision legacy Fortran 77, 66, or even Fortran IV.

In such cases, analytic derivatives are not available, and must be estimated. However, the level of error incurred must be considered. Unfortunately, the subject of finite-difference estimates of derivatives is largely absent from the applied mathematics literature. Researchers are typically interested in issues such as convergence rates, radii of convergence, and breadth of applicability of methods. Little attention is paid to the accuracy of results compared to available precision. Numerical experiments are typically performed in double precision, and results reported to between five and seven significant figures. When derivatives are needed, they are usually obtained via forward differencing (to keep the total number of function evaluations low) with some fixed step size. The availability of 16 figures of accuracy makes the error associated with this method acceptable.

In this industry, however, we often don't have the luxury of throwing away half of



our significant digits, especially when forced to work with single-precision code. In this article, I'll implement techniques for dealing with this situation, and present the results of numerical experiments.

I encountered the particular development problem that prompted this article while working on a software package that predicts the performance of compressors. Typically, users adjust various parameters (such as RPM, amount of fluid moving through the compressor, or the physical dimensions of the part) in an attempt to achieve peak performance while still meeting design criteria for weight, overall size, and so on.

In an attempt to automate some of the simpler problems that the designer was solving by hand, we hooked the package up to a constrained optimizer. The idea was to have the optimizer vary selected inputs to the package to maximize performance while still satisfying various constraints. One of the challenges we needed to overcome was the lack of analytic derivatives. The application was already fairly complex and solved many of its subproblems iteratively. The code would need to be treated as a black box, forcing us to approximate the derivatives numerically. Further complicating things, the core analysis routines were made up of tens of thousands of lines of Fortran 77 written in single precision. That is, we were beginning with only about seven digits of accuracy, some of which were lost during the analysis. The additional loss of accuracy incurred by naive use of forward differences essentially prevented the optimizer from converging to any real accuracy.

Theoretical Basis

There are two primary sources of error in all finite-difference estimates of derivatives:

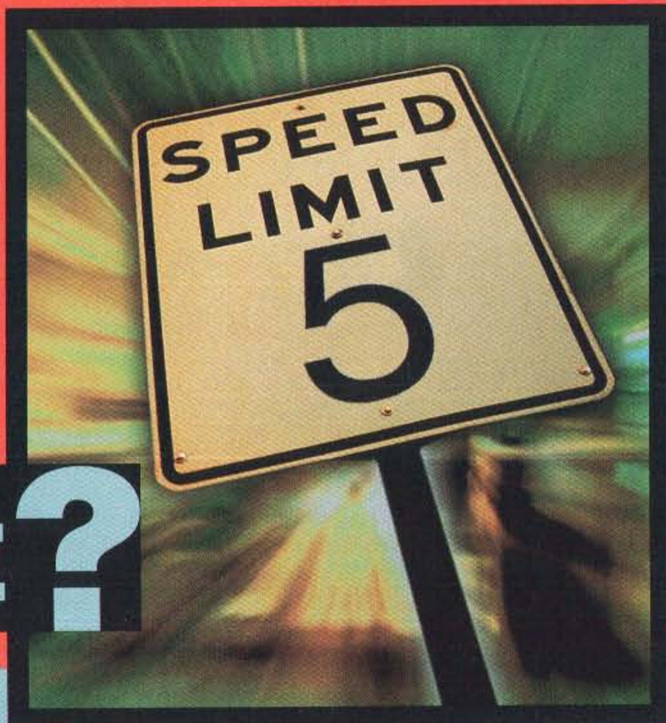
- Truncation error, stemming from our having thrown away higher order terms in the Taylor series.
- Condition error, stemming from the finite-precision available on a computer.

I'll turn first to the truncation error in forward differences. (For more information

Michael is a C++ developer at Northern Research and Engineering Corp. He can be reached at mgh@world.std.com.

IT'S A CODE PERFORMANCE PROBLEM.

BUT WHERE?



ONLY NuMEGA TrueTime™ PROVIDES SEAMLESS MULTI-LANGUAGE PERFORMANCE ANALYSIS FOR COMPONENT-BASED APPLICATIONS.

Now, for the first time, you can easily and accurately pinpoint code performance bottlenecks no matter what language they are written in or where they reside. In Visual Basic, Visual C++, and Java, NuMega TrueTime lets you see exactly where problems occur. Get NuMega TrueTime and get all your applications and components up to speed.

Microsoft Enterprise Development Partner

Compuware® NuMEGA®

1-888-686-3430 www.numega.com

SmartDebugging™ for faster development VISUAL BASIC® VISUAL C++® JAVA™

30-DAY MONEY BACK GUARANTEE

Copyright©1998. Compuware Corporation. All rights reserved. All trademarks and registered trademarks are the property of their respective owners.

(continued from page 52)
on this topic, see *Practical Optimization*, by Philip Gill et al., Academic Press, 1981, ISBN 0-1-228-3952-8.)

For more information on Taylor's theorem (Example 1(a)), turn to any standard work on calculus. Solving for df/dx leads immediately to the forward difference formula in Example 1(b) for estimating the first derivative. The second term on the right, which is dropped in making the estimate, is referred to as the "truncation error" (because we are truncating the Taylor series for f). If the calculations were being performed to infinite precision, that would be the end of the story—smaller step sizes give smaller errors. Of course, that's not the case. Since f is being evaluated on a computer to a fixed precision, there is some approximation error. Let $g(x)$ be the computed value of $f(x)$. That is, $g(x)=f(x)+eps(x)$ (so that $eps(x)$ is the floating-point error in computing f at x on a computer). If f is a simple function, *eps* would typically be bounded by $|f(x)|eps(M)$, where $eps(M)$ is the machine precision ($2^{-23}\approx10^{-7}$ for floats, $2^{-53}\approx10^{-16}$ for doubles). However, for more complex functions, $eps(x)$ may be considerably larger. Taking this into account, Example 1(c) is the actual estimate. The last term, the error term, is referred to as "condition error."

You can now see how things get interesting. Truncation error gets smaller as the step size decreases, but the condition error grows. Suppose you can bound

$$\frac{d^2f}{dx^2}(x)$$

by some constant, say M (at least near x). Suppose too, that you can bound $|eps(x)|$ by another constant, E, near x. I'll denote $(g(x+h)-g(x))/h$ by $\phi(x)$ (in other words, $\phi(x)$ will be the computed estimate of the derivative); then $|\phi(x)-df/dx(x)|\le Mh/2+2E/h$.

Clearly then, there exists an optimal value for h; the first term on the right increases with h, while the second term decreases with it. To find that optimum, you differentiate the right side with respect to h to obtain Example 1(d). If you set this equal to zero and solve for h, you get Example 1(e).

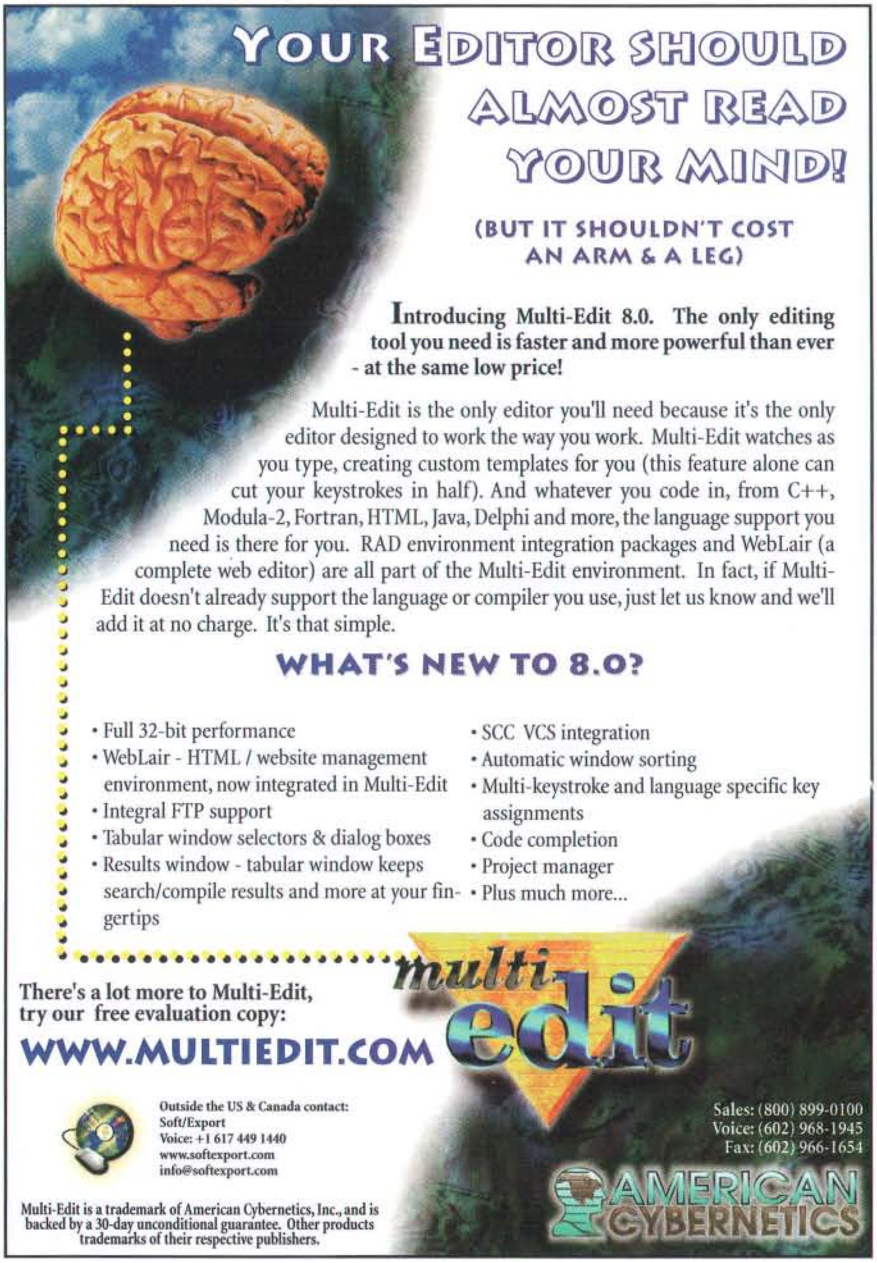
When the step size is to be fixed (this is usually done for performance reasons), the square root of the machine precision is often used. An identical analysis yields the following step sizes for other formulas:

- Example 2(a) shows the central difference approximation to the first derivative, where in this case M is an upper bound on d^3f/dx^3.
- Example 2(b) is the central difference approximation to the second derivative, where in this case M is an upper bound on d^4f/dx^4.

If you do have analytic derivatives available, the aforementioned observations can be turned around to estimate E, the maximal error in evaluating f. Recall the bound on the error in forward differencing: $|\phi(x)-df/dx(x)|\le Mh/2+2E/h$. Notice that if h is small enough, the error bound will be dominated by the second term on the right side. Under this assumption, you have $|\phi(x)-df/dx(x)|\le 2E/h$, or $E=O(|\phi(x)-df/dx(x)|h)$.

In other words, assume you know the correct value of $df/dx(x)$. You can intentionally induce error in the forward difference approximation to $df/dx(x)$ by intentionally choosing a very small step size. This will cause the condition error to balloon up, and the truncation error to die off, so you can drop the first term in our upper bound on $|\phi_f-df/dx(x)|$. Then, since $df/dx(x)$ is known, you can solve for E, and so estimate how much floating-point error is creeping into your function evaluations.

For this method to give realistic estimates in practice, a few subtle points need to be considered. The first is that the quantity $2E/h$ is only an upper bound on the finite-difference error. There is no guarantee that the actual error for a given step size will hit this upper bound. In fact, it is often the case that the actual error in evaluating a function varies widely at



AD LINK 143

adjacent points. The second concerns the choice of step size. If it is taken to be too small, the forward difference will be zero, and any information on the error will have been lost. If it is taken to be too large, the assumption that the error bound is dominated by condition error will no longer be valid. Lastly, what if analytic derivatives are not available? These concerns are addressed in algorithms that implement these ideas.

Algorithms

I will now present algorithms for high-precision estimates of first derivatives through forward differencing (see the aforementioned *Practical Optimization*). The idea is to get a reasonable estimate of M (the upper bound on f's second derivative) through straightforward central difference estimates. The step size for forward differencing is then chosen to be Example 3(a).

The bulk of the work, then, is to determine a step size giving a suitable estimate of M. I assume that the second derivative of f is changing slowly enough that its value at x is an acceptable approximation to M. A sequence of trial values for that step size is generated, and at each point, the relative condition error is calculated as in Example 3(b). Here, E is our bound on the evaluation error in f, h is the trial step size, and Φ is the estimate of f's second derivative. $condErr(\Phi)$ is a measure of the size of the condition error only (that is, neglecting truncation error) in estimating the second derivative, relative to the size of the second derivative. Likewise, the relative condition errors of the forward and backward difference estimates of f's first derivative are found from:

$$condErr\ (\phi_f) = \frac{2E}{h\,|\phi_f|}$$

$$condErr\ (\phi_b) = \frac{2E}{h\,|\phi_b|}$$

For a value of h to be considered acceptable, $condErr(\Phi)$ must be less than 0.1 (guaranteeing at least one correct significant digit, at least with respect to the condition error). To carefully assess the accuracy of Φ, some estimate of the truncation error is needed. However, it is inconvenient to do this, and so instead, the relative condition error is required to be at least 0.001, since the truncation error generally rises as the condition error falls. To obtain an initial estimate of h, the assumption shown in Example 3(c) is made.

In other words, it is assumed that f's second derivative is proportional to f, and inversely to the square of its argument (note that this assumption is satisfied by any monomial of degree greater than 1). The "1s" in both numerator and denominator guard against the effects of small values for either f or x. If this assumption holds, then, from Example 1(e), the optimal interval is Example 3(d).

However, at a step size of $\bar{h}$, $condErr(\Phi)$ will be of order one, so we choose as our initial estimate of h the quantity $10\bar{h}$ to produce a $condErr(\Phi)$ of 0.01. Of course, it is entirely possible that the assumption on the behavior of f's second derivative is incorrect. If the initial estimate of Φ produces a relative condition error that is too small, you simply decrease h, and if $condErr(\Phi)$ is too large, h is increased. This is repeated until an acceptable value is found.

Notice that this iteration could fail to ever produce an acceptable value of h if, say, f is constant. For this reason, we set an upper limit on the number of iterations, k_{max}. For diagnostic purposes on failure, we also record whether or not we ever found a value of h that was acceptable for estimating the first derivative. This information is encoded into the variable h, which is initialized to –1, and set to the step size h the first time we find an h for which both ϕ_f and ϕ_b have relative condition errors of less than 0.1. (In the following algorithms, ϕ_f denotes the forward difference approximation to the derivative, ϕ_b the backward difference approximation, and ϕ_c the central difference approximation.)

AD LINK 405

$$\text{(a)}\quad f(x+h)=f(x)+\frac{df}{dx}(x)h+\frac{1}{2}\frac{d^2f}{dx^2}(\zeta)h^2,\ \text{where } x\le\zeta\le x+h$$

$$\text{(b)}\quad \frac{df}{dx}(x)=\frac{f(x+h)-f(x)}{h}-\frac{1}{2}\frac{d^2f}{dx^2}(\zeta)h$$

$$\text{(c)}\quad \frac{g(x+h)-g(x)}{h}=\frac{f(x+h)+eps(x+h)-f(x)-eps(x)}{h}=\frac{f(x+h)-f(x)}{h}+\frac{eps(x+h)-eps(x)}{h}$$

$$\text{(d)}\quad \frac{1}{2}(M)-2Eh^2$$

$$\text{(e)}\quad h_{Opt}=2\sqrt{\frac{E}{M}}$$

***Example 1:** (a) Taylor's theorem; (b) solving for* df/dx; *(c) actual estimate; (d) computed estimate; (e) solving for* h.

$$\text{(a)}\quad \frac{df}{dx}(x)\approx\frac{f(x+h)-f(x-h)}{2h},\ \text{and } h_{Opt}=\sqrt[3]{3\frac{E}{M}}$$

$$\text{(b)}\quad \frac{d^2f}{dx^2}(x)\approx\frac{f(x+h)-2f(x)+f(x-h)}{h^2},\ \text{and } h_{Opt}=\sqrt[2]{3\frac{E}{M}}$$

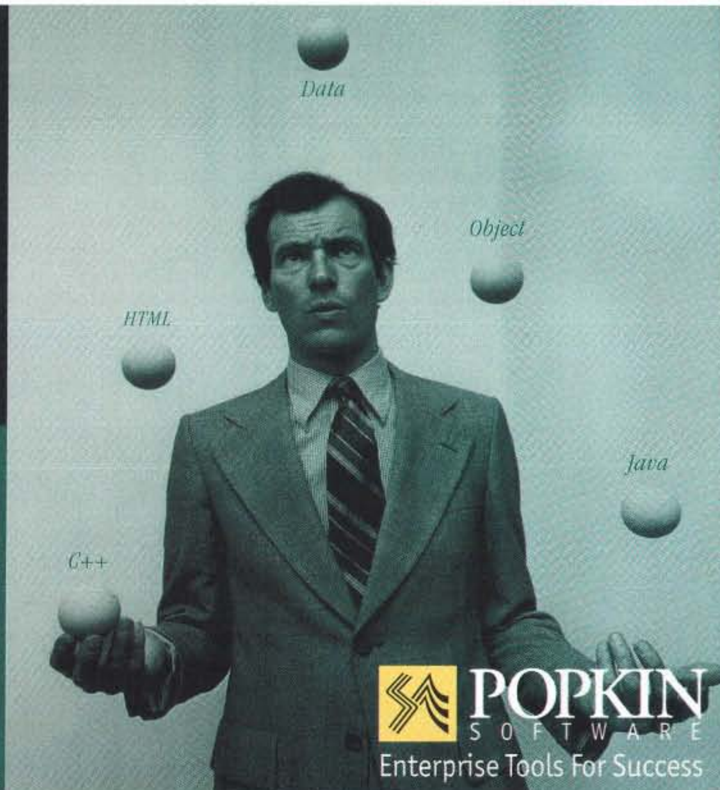
***Example 2:** (a) Difference approximation to the first derivative; (b) difference approximation to the second derivative.*

Algorithm #1

1. Initialization. Set $h=10\bar{h}$, where $\bar{h}=2(1+|x|)\sqrt{(E/(1+|f(x)|)}$. Set $k=0$. Evaluate $\phi_f(h)$, $\phi_b(h)$, $\Phi(h)$, and their relative condition errors. Set $h_s=-1$.
2. Evaluate the initial estimate. If $max(condErr(\phi_f), condErr(\phi_b)\le 0.1$, set $h_s=h$. If $0.001\le condErr(\Phi)\le 0.1$, go to step 5. If $condErr(\Phi)<0.001$, go to step 4. Otherwise, continue at step 3.
3. Relative condition error too large.
   a. Set $k=k+1$, and $h=10h$.
   b. Recompute the finite differences and their relative condition errors.
   c. If $h_s=-1$, and $max(condErr(\phi_f), condErr(\phi_b))\le 0.1$, set $h_s=h$.
   d. If $condErr(\Phi)\le 0.1$, go to step 5.
   e. If $k=k_{max}$, go to step 6.
   f. Go back to step 3a.
4. Relative condition error too small.
   a. Set $k=k+1$, and $h=h/10$.
   b. Recompute the finite differences and their relative condition errors.
   c. If $h_s=-1$, and $max(condErr(\Phi_f), condErr(\Phi_b))\le 0.1$, set $h_s=h$.
   d. If $condErr(\Phi)\ge 0.1$, go to step 5.
   e. If $k=k_{Max}$, go to step 6.
   f. Go back to step 4a.
5. Success. Set $h_{Opt}=2\sqrt{E/|\Phi|}$, the error estimate is 0.

"I don't want to juggle a bunch of stand alone modeling tools anymore. All I want is one integrated family"

In enterprise modeling you can't afford to stand alone—or use stand alone products. Only the System Architect suite of tools allows you to share Business Processes, Objects and Data throughout your development team. We give you full multi-user networking capabilities. As one team member makes a change, the repository is automatically updated for all other members. Popkin has over 10 years of experience to meet your needs now, and far into the future. So stop the juggling act. Try us FREE for 30 days!



POPKIN SOFTWARE
Enterprise Tools For Success

You want it? You got it! FREE System Architect Eval : 800.732.5227

Argentina 54-1-328-0324 * Australia 61-3-6234-6499 * Benelux 31-30-666-55-30 * Brazil 55-11-5181-0246 * Chile 56-2-633-4733 * France 33-389-444476 * Israel 972-9-9-598-999 * Italy 39-49-8283411 * Korea 212-989-9809 * Malaysia 60-3-757-1806 * Mexico 525-531-6768 Spain 34-3-415-7800 * Sweden 46-8-626-8100 * Switzerland 41-61-6922666 * Turkey 90-212-257-48-70 * United Kingdom, Ireland, Denmark, Poland: 44-1926-450858 * ©1997 Popkin Software & Systems. The System Architect logo is a trademark of Popkin Software & Systems. All other brand and product names are trademarks or registered trademarks of their respective holders. Popkin Software & Systems, 11 Park Place, New York, NY 10007, Phone: 212.571.3434, Fax: 212.571.3436

AD LINK 118

6. Failure.
   a. If h_s=1, then f appears to be nearly constant. Set h_{Opt}=h, and set ϕ_f=0; the error estimate is zero.
   b. If $h_s \neq -1$ and *condErr*(Φ)>0.1, then f appears to be linear or odd. Set h_{Opt}=h_s, Φ=0, and set the error estimate as in step 5.
   c. Otherwise, the second derivative appears to be changing too rapidly. Set h_{Opt}=h and set the error estimate as in step 5.

Algorithm #1 assumes the availability of E. Often this is not a problem; E may be estimated by $E \sim |f|$*(machine precision)*. In other cases, however, this may not be sufficient. If analytic derivatives are available, then the method outlined here could be used to compute *eps(x)*. Of course, if you have analytic derivatives, then you don't care about Algorithm #1 (although I suppose that if derivative evaluation were expensive enough, you might do it once to get *eps*, and then use Algorithm #1). In general, though, the problem is that initially, you have neither a reasonable estimate of the derivative or of E. Luckily, I've found that, in practice, an order of magnitude estimate of E is sufficient for Algorithm #1. Consequently, Algorithm #2 may be used to perform a high-precision estimate of the derivative when only function evaluations are possible.

Algorithm #2

1. Use some fixed step length to estimate *df/dx* through forward differences.
2. Use this result to estimate E (see Algorithm #3).
3. Use the result of step 2 in Algorithm #1.

Previously, I mentioned that there were a few subtle points to keep in mind when trying to back out E. Algorithm #1 takes them into account. The idea is to find the smallest step size that produces a change in the function value, and use that for a forward difference estimate. Any step size less than that gives no useful information, since the resulting finite-difference estimate will always be zero, independent of the function. On the other hand, the smaller the step becomes, the more the condition error term dominates the error bound. This leads you to use the smallest step size possible.

The algorithm proceeds in three distinct phases. The first attempts to bracket the least significant step; A denotes the lower bound and B the upper. The second phase uses a simple bisection algorithm to find the least-significant step within a user-specified relative tolerance, *relTol* (this does not have to be unduly tight; in practice; I've found that a value of 0.1 works well). The last phase takes this approximation to the least-significant step and uses it to compute the forward, backward, and central difference approximations to the first derivative. These are then compared to the "known" derivative value, and (for each approximation) E is computed. The maximum value is returned.

Algorithm #3

1. Bracketing Phase. Break the procedure into cases: If both $f(x)$ and $df/dx(x)$ are nonzero, go to step 2. If $df/dx(x)$ is nonzero, but $f(x)$=0, go to step 3, and if both $f(x)$ and $df/dx(x)$ are zero, go to step 4.

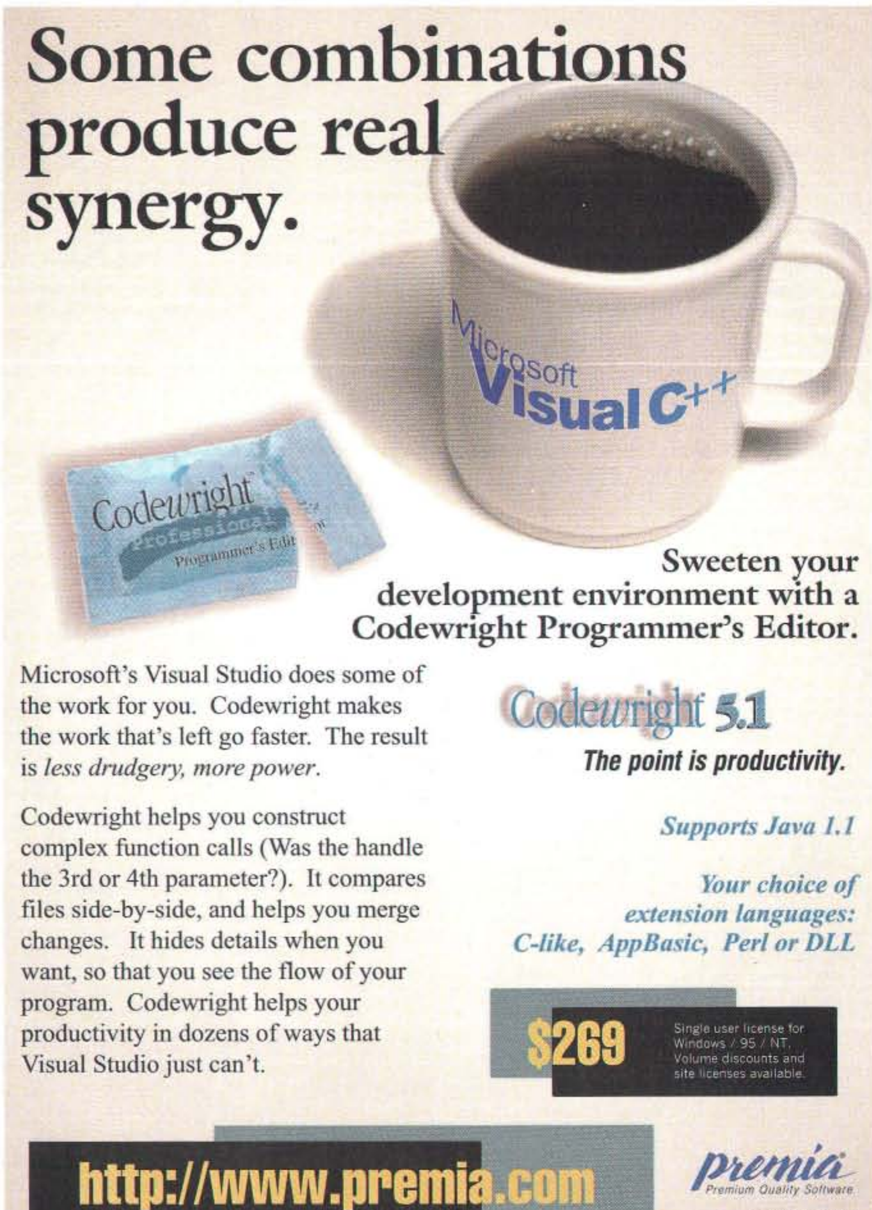
(a) $$h_{Opt}=2\sqrt{\frac{E}{M}}$$

(b) $$condErr(\Phi)=4\,\frac{E}{h^2|\Phi|}$$

(c) $$o\left(\frac{d^2f}{dx^2}(x)\right)=o\left(\frac{1+|f(x)|}{(1+|x|)^2}\right)$$

(d) $$\bar{h}=2(1+|x|)\sqrt{\frac{E}{(1+|f(x)|)}}$$

Example 3: *(a) Step size for forward differencing; (b) condition error in estimating the second derivative is calculated; (c) obtaining an initial estimate of* h; *(d) optimal interval.*



AD LINK 121

$$f(x,y)=\begin{pmatrix} 2\sin(x)-\cos(y)+\cos(x) \\ -2-\cos(x)+2\sin(y)+3\cos(y) \end{pmatrix}$$

***Figure 1:** Test 1: The trigonometric function.*

2. Bracketing Phase, Case 1. A reasonable guess for h might seem to be $|x|eps(M)$, which would be the least-significant change you could possible make in x. However, when x is small, this can lead to absurdly small guesses for h, which slow the algorithm considerably. So, when x is small, we instead reason that $f(x)+eps(M)|f(x)|\sim f(x+h)\sim f(x)+df/dx(x)h$ when h is the smallest significant step. So, if $|x|\geq 1$, let $A=|x|eps(M)$ else let $A=eps(M)|f(x)|\backslash|df/dx(x)|$. Let B=10A.
   a. Ensure that A is a Lower Bound. While $f(x+A)\neq f(x)$, let $B=A$, $A=A/10$.
   b. Ensure that B is an Upper Bound. While $f(x+B)=f(x)$, let $A=B$, B=10B.
   c. Go to step 5.
3. Bracketing Phase, Case 2. Since $f(x)=0$, function values near x are very small (near underflow), and the smallest-significant step size may be hard to find. Instead, we simply set A to 0.
   a. Ensure that B is an Upper Bound. While $f(x+B)=f(x)$, let $A=B$, B=10B.
   b. Go to step 5.
4. Bracketing Phase, Case 3. Since $df/dx(x)=0$, the function may even be constant, so we must set some maximum number of guesses, say k_{max}, let $A=0$, $k=0$.
   a. Ensure that B is an Upper Bound. While $f(x+B)=0$, and $k\leq k_{max}$, let $A=B$, B=10B, $k=k+1$.
   b. If $k=k_{max}$, then the function appears to be constant—return an error code.
   c. Go to step 5.

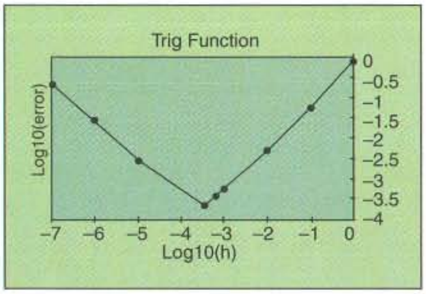


***Figure 2:** A log-log plot of Test 1's step sizes against errors.*

5. Search Phase. Carry out a binary search until A and B agree to *relTol*.
6. Estimation Phase. Compute $\phi_f(B)$, $\phi_b(B)$, $\phi_c(B)$. Compute $err(f)=|\phi_f(B)-df/dx(x)$, $err(b)=|\phi_b(B)-df/dx(x)|$, and $err(c)=2|\phi_c(B)-df/dx(x)|$.
7. Return $max(err(f), err(b), err(c))$.

Numerical Experiments

I've implemented these algorithms in C++ (available electronically; see "Resource Center," page 3), and applied them to two test functions that show the algorithms in their best and worst light. In each case, the derivative was first estimated cheaply (comparatively) in the following way:

1. Estimate E by $(1+|f(x)|)eps(M)$.
2. Set $h=|x|\sqrt{(E)}$.
3. Use h to compute ϕ_f.

That estimate was then used in Algorithm #2 to produce a more careful approximation.

Figure 1 presents the first test function. Table 1 lists the errors incurred by forward difference approximations to the first partial derivative with respect to x at the point (0,0) (the exact value is -2). Figure 2 shows a log-log plot of the step sizes against the errors.

Step Size	Estimated Derivative	Log10(step)	Log10(error)
1.000000E–07	1.78814	–7	–0.673951032
1.000000E–06	2.02656	–6	–1.575771929
1.000000E–05	2.00271	–5	–2.567030709
3.452670E–04	1.99979	–3.461844929	–3.677780705
6.406720E–04	1.99964	–3.193364256	–3.443697499
1.000000E–03	1.99947	–3	–3.27572413
1.000000E–02	1.99497	–2	–2.298432015
1.000000E–01	1.94671	–1	–1.27335428
1.000000E+00	1.22324	0	–0.109713147

***Table 1:** Errors incurred in Test 1.*

Step Size	Estimated Derivative	Log10(step)	Log10(error)
1.000000E–04	1.144410E–01	–4	–2.121667819
1.000000E–03	1.220700E–01	–3	–4.140861703
2.444270E–03	1.242680E–01	–2.611850823	–2.643916751
1.000000E–02	1.319410E–01	–2	–2.002469457
1.035800E–01	2.255080E–01	–0.984724093	–0.985016433
1.000000E+00	1.121400E+00	0	–0.000259655

***Table 2:** Errors incurred in Test 2.*

$$f(x)=(x-100)^2+10^{-6}(x-300)^3$$

***Figure 3:** Test 2: Example 8.5 from* Practical Optimizations.

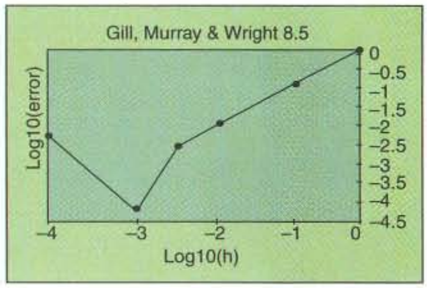


***Figure 4:** A log-log plot of Test 2's step sizes against errors.*

Clearly, the optimal step length is near 3.45266e–4, which happens to be the square root of machine precision. So, in this case, the standard step length does quite well, giving an estimate of about 1.99979. Algorithm #1 yields a step size of 6.40672e–4, and yields an estimate of about 1.99964, comparable in accuracy to the standard method.

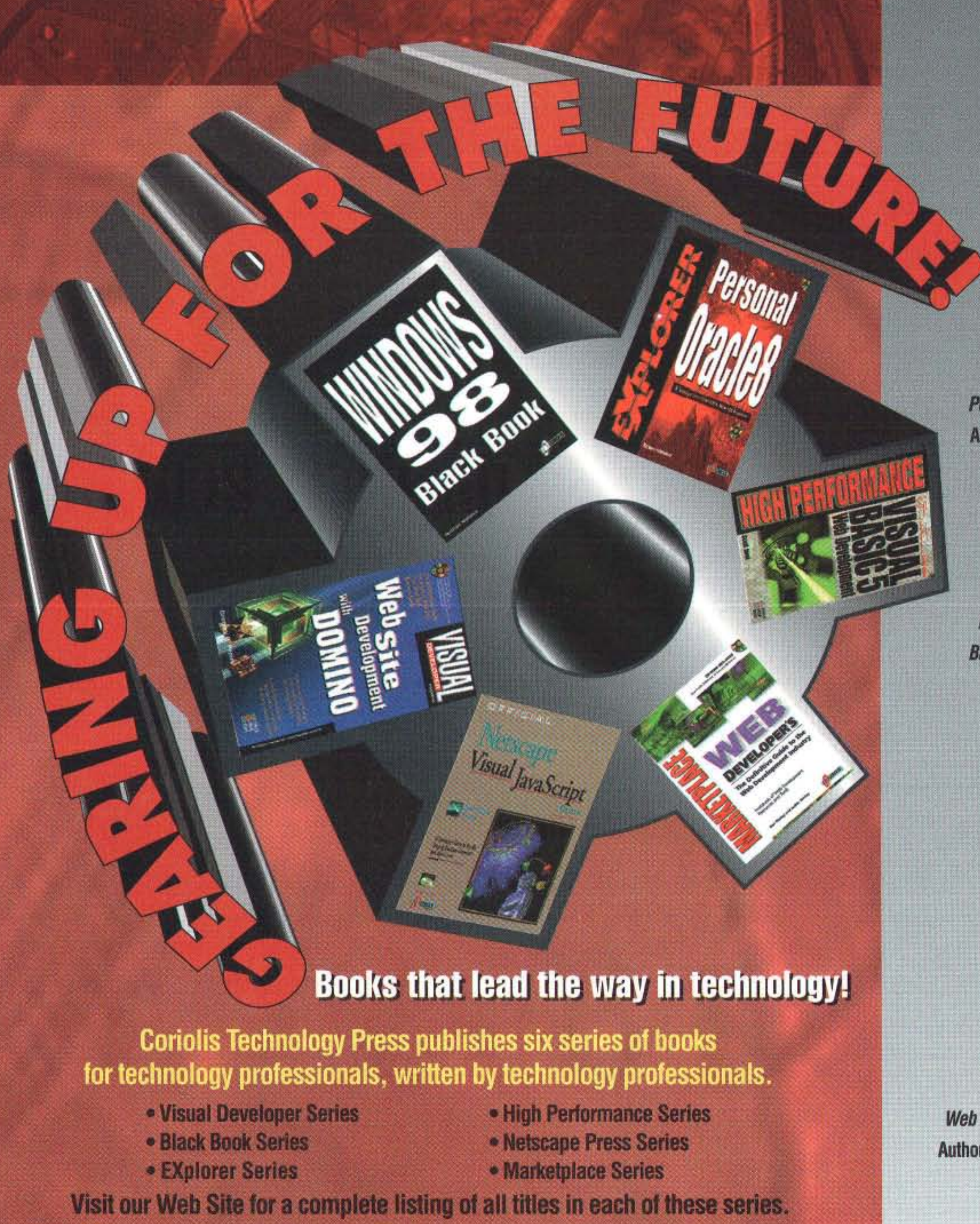
Figure 3 presents the second test function (based on Example 8.5 in *Practical Optimization*) Table 2 lists the errors incurred in estimating the derivative at x=100.001 (the exact value is about .1219977). Figure 4 displays the same log-log plot as in Figure 2. In this case, the quick-and-dirty approach yields a step length of about .10358, and an estimated derivative of .225505 (or no correct significant digits). Algorithm #1, on the other hand, yields a step length of about 2.44427e–3, and an estimated derivative of about .124268 (or two correct significant digits).

Conclusion

The algorithms presented here can certainly improve the accuracy of numerical methods, though at the cost of significantly more function evaluations. They can principally be applied to practical situations where obtaining greater accuracy from the function evaluations is difficult or impossible. Another application of estimates of function evaluation error lies in appropriately tolerancing iterative schemes. Again, this generally is not covered in the academic literature, where convergence criteria can be adjusted as needed. However, in engineering software development, solvers must work reliably over a wide range of problems, and a tolerancing scheme that adapts to the available accuracy can help.

DDJ

CORIOLIS TECHNOLOGY PRESS©



Books that lead the way in technology!

Coriolis Technology Press publishes six series of books for technology professionals, written by technology professionals.

- Visual Developer Series
- Black Book Series
- EXplorer Series
- High Performance Series
- Netscape Press Series
- Marketplace Series

Visit our Web Site for a complete listing of all titles in each of these series.

So gear up for the future with publications from professionals. Gear up with Coriolis Technology Press!

1.800.410.0192 • Int'l Callers: 602.483.0192
Fax: 602.483.0193 • www.coriolis.com

Available at Bookstores and Computer Stores Nationwide

Prices and availability dates are subject to change.
© 1998 by Coriolis Technology Press. All Rights Reserved. CP/AS - MDDJ0598

Visual Developer Web Site Development with Domino
Author: Gregory Pepus
Pages: 500
ISBN: 1-57610-172-X
Price: $39.99 U.S.
$55.99 CANADA

Windows 98 Black Book
Author: Nathan Wallace
Pages: 600
ISBN: 1-57610-265-3
Price: $49.99 U.S.
$69.99 CANADA

Personal Oracle8 EXplorer
Author: Richard Fieldhouse
Pages: 650
ISBN: 1-57610-250-5
Price: $49.99 U.S.
$69.99 CANADA

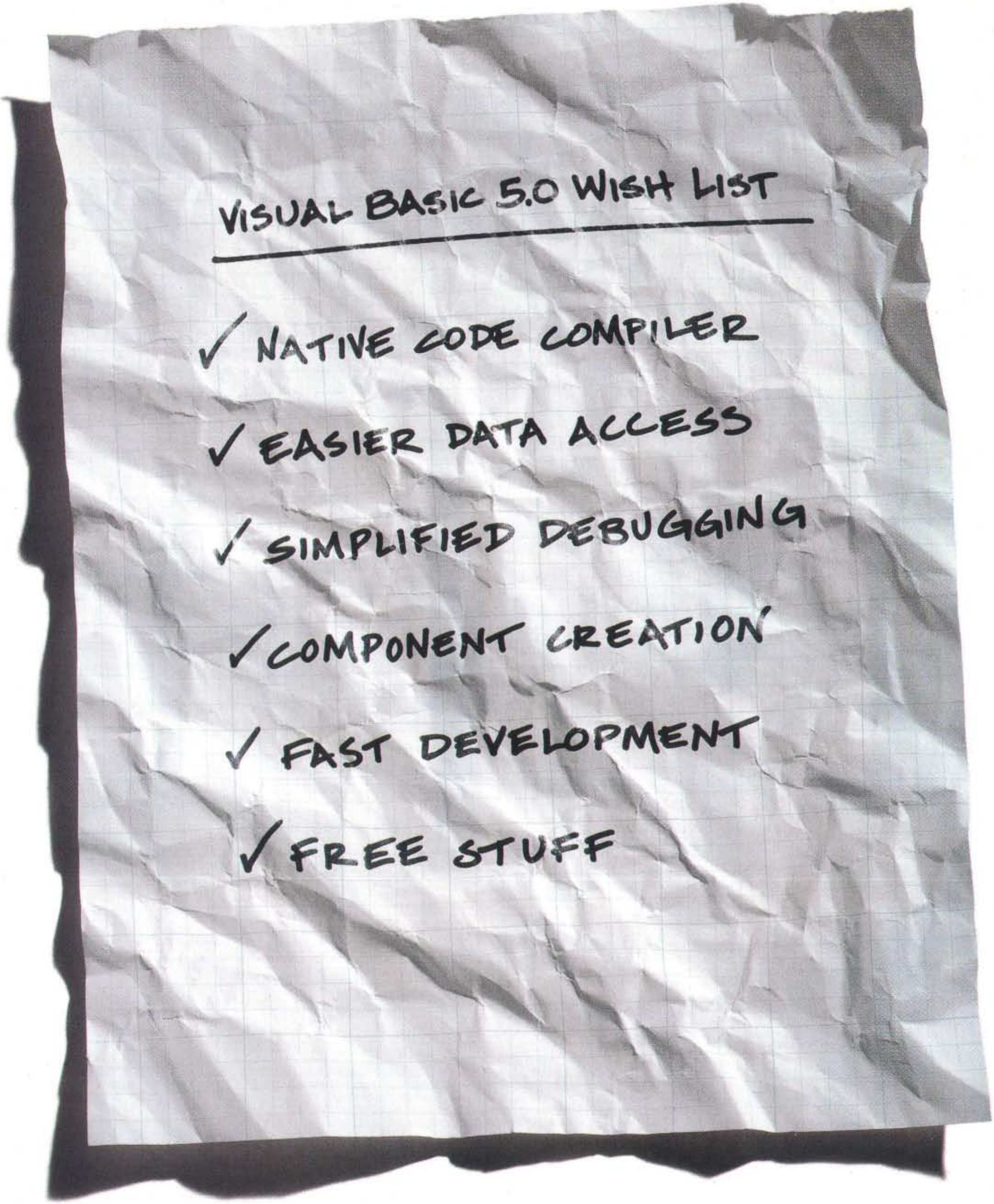
High Performance Visual Basic 5 Web Development
Author: Scott Jarol
Pages: 480
ISBN: 1-57610-063-4
Price: $39.99 U.S.
$55.99 CANADA

Official Netscape Visual JavaScript Book
Author: Doug Lloyd
Pages: 450
ISBN: 1-56604-761-7
Price: $49.99 U.S.
$69.99 CANADA

Web Developer's Marketplace
Authors: Dan and Judith Wesley
Pages: 700
ISBN: 1-57610-178-9
Price: $49.99 U.S.
$69.99 CANADA

CORIOLIS

AD LINK 423



*Developer Edition has a limited license for development and testing purposes only. © 1998 Microsoft Corporation. All rights reserved. Microsoft, *Where do you want to*

Microsoft®
VISUAL
Studio
Visual Basic®
Visual C++®
Visual InterDev™
Visual J++™
Visual FoxPro™

You asked for it, you got it.

(Who says we're not easy to work with?)

The Microsoft® Visual Basic® 5.0 development system has everything you'd want in an easy-to-use RAD development tool. Not surprisingly, over 1 million developers have already discovered its speed, power and ease-of-use. *PC Magazine* named it "Best of 1997." *InformationWeek* put it on its list of "Most Important Products." And *PC Computing* made it a 1997 MVP Award winner. What's more, pick up your copy of Visual Basic 5.0 Professional Edition today and you'll also get a copy of Microsoft Windows NT® Workstation Developer Edition 4.0* absolutely free. For more information see your local reseller or go to www.microsoft.com/developer/offer/

Where do you want to go today?® Microsoft®

go today?, Visual Basic, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The Pentium F00F Bug

Workarounds for a nasty problem

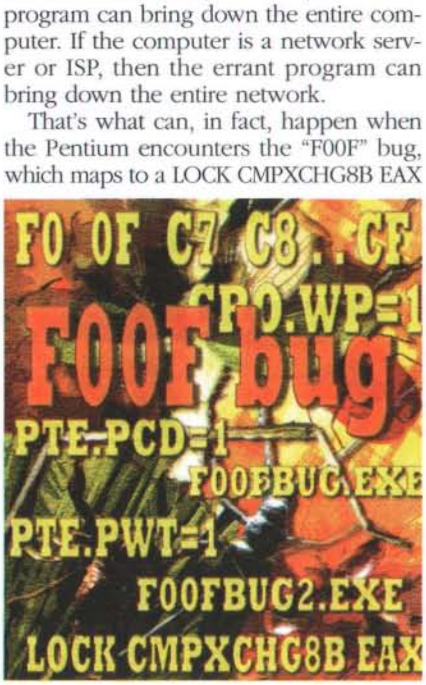
Robert R. Collins

Last fall, a message warning users of a bug in the Pentium/Pentium MMX was anonymously posted on the comp.os.linux.advocacy Internet newsgroup. According to the message, the bug could completely lock up the computer from any operating mode in any operating system. At first glance, the ordinary news reader might say "so what." After all, many users are accustomed to Windows 3.1/95 regularly locking up. But the placement of the bug on comp.os.linux.advocacy was calculated and intentional. The readers on that newsgroup knew exactly what the bug meant—an ordinary user or saboteur could unleash a program to bring down network servers that form the backbone of our modern networking community. Internet service providers (ISPs), web hosts, government agencies, and computer departments at universities were petrified of the potential damage that this bug could do.

Enter the F00F Bug

When any x86 processor from the 80186 and beyond encounters an invalid instruction, the processor is *supposed* to generate an invalid opcode exception. In Intel vernacular, the undefined opcode exception is known as a "#UD." This handler usually signals an error condition and terminates the errant program. When this mechanism works, the errant program can't harm the computer system. Should this mechanism fail, however, the errant program can bring down the entire computer. If the computer is a network server or ISP, then the errant program can bring down the entire network.

That's what can, in fact, happen when the Pentium encounters the "F00F" bug, which maps to a LOCK CMPXCHG8B EAX



instruction. CMPXCHG8B compares 64-bit memory contents with the contents in EDX and EAX. One of the operands must be memory, and the other (implied) operand is EDX:EAX. It is possible to construct an instruction encoding that doesn't map to a memory operand. Since the nonmemory form of this instruction is invalid, a compiler or assembler will not generate this code. Instead, assembly-language programmers must construct it by hand.

Such an illegal encoding should generate the requisite #UD. As you'd expect, a CMPXCHG8B EAX instruction generates a #UD. However, when this illegal encoding is prepended with a LOCK prefix, the processor fails to work correctly.

Using the LOCK prefix on this form of CMPXCHG8B is illegal in and of itself. LOCK prefixes are only allowed on memory-based read-modify-write instructions. Hence a LOCK prefix on the register-based CMPXCHG8B EAX instruction should also generate an invalid opcode exception.

Instead, the Pentium locks up and freezes the entire computer when it encounters this instruction. This bug is especially nasty, because any user can construct a program with this instruction, and upload it to a network computer, or incorporate it within an ActiveX applet. Once the program is run on the network, the network server crashes. The only possible recovery comes by hitting the big red switch. Suppose you download an ActiveX applet that contains this code. As soon as the code executes, *your* computer freezes up.

Within one week of the discovery of the bug, Intel announced a software workaround that can be incorporated into virtually any operating system (except real-mode operating systems, like DOS).

How the Bug Works

When the processor encounters the instruction F0 0F C7 C8 (or anything from F0 0F C7 C8..CF), the F00F bug occurs. The processor recognizes that an invalid opcode has occurred and tries to dispatch the #UD handler. Because of the LOCK prefix, the processor is confused. When the processor issues the bus reads to get the #UD handler vector address, the processor erroneously asserts the LOCK# signal. The LOCK# signal can only be asserted for read-modify-write instructions that modify memory. When the bus is locked, a locked memory read must be followed by a locked memory write, lest

Robert is an independent consultant on the x86 architecture. He can be reached at rcollins@x86.org.

Rogue Wave SOFTWARE

Introducing

Tools.h++ Professional

Essential Tools for C++ Professionals

For years, **Tools.h++** has provided C++ developers like you with rich, robust, and versatile foundation classes useful for building virtually any application. Today, your applications must perform in more complicated hardware and software environments than ever before.

Now, **Tools.h++ Professional** gives you the expanded and integrated tool set you need to meet the challenges of even your most complex development project.

Tools.h++ Professional includes Tools.h++, with over 130 foundation classes, plus Java/C++ interoperability, networking, and CORBA tools. Finally, a collection of powerful C++ tools that help you quickly build solid, portable applications for today's complex computing conditions.

Tools.h++ Professional. Tested and proven code, built by C++ language experts. Easy portability to most popular platforms. Network computing and language interoperability solutions, plus fundamental C++ building blocks. A wealth of functionality in one convenient and affordable toolbox.

What are you waiting for?

Tools.h++ Professional gives you:

Tools.h++
- **Fundamental C++ building blocks**
- **Easy-to-use interface to Standard C++ Library**

Networking Tools
- **Network communication classes**
- **Thread-hot Internet classes**

Java/C++ Interoperability Tools
- **Java™ virtual streams**
- **C++ implementation of the Java serialization format**

CORBA Tools
- **CORBA streaming classes**

Tools.h++ Professional contains technology formerly sold separately as Serialize.h++, JTools, Net.h++, Inter.Net.h++, and ORBstreams.h++.

www.roguewave.com/ad/best

Call us toll free in the U.S. at (800) 487-3217.

Telephone us in Europe: Rogue Wave Software GmbH: +49-6103-59 34-0 ◆ Rogue Wave Software B.V.: +31-20-416 06 57

Rogue Wave Software S.A.R.L.: +33-1-5568 1008 ◆ Rogue Wave Software-UK. Ltd.: +44-118-988-0224

Rogue Wave and .h++ are registered trademarks of Rogue Wave Software, Inc. Java is a trademark of Sun Microsystems, Inc. All other trademarks are the property of Rogue Wave Software, Inc. or their respective holders.

AD LINK 425

(continued from page 62)

unpredictable results may occur. But in this case, the LOCK# signal remains asserted for the two consecutive memory reads required to retrieve the #UD vector address. The processor never issues any intervening locked write, and then hangs itself. This behavior is shown in the logic analyzer trace in Example 1. As you can see, the Pentium tries to retrieve the #UD vector with two locked reads. After that, all processor activity stops.

```
Sequence  Address    Data      Mnemonic                          Timestamp
-------------------------------------------------------------------------
T 524285  000000B2----7E--------(-IO-WRITE-)--------------------300 ns
  524286  00000018  ----E14C   ( LOCKED MEM READ )               440 ns
  524287  0000001A  F000----   ( LOCKED MEM READ )               100 ns
```

Example 1: *F00F bug example.*

The Various Workarounds

There are various workarounds to this bug—not all of them are good (a few are outright kludgey, in fact). One workaround Intel has proposed actually takes advantage of the bug's behavior to do the right thing. Another Intel workaround is ingenious, though a kludge. The first two alternate workarounds, to be presented shortly, are given for academic purposes only. Even though the workarounds have demonstrated their ability to obscure the bug behavior, they are not entirely reliable.

Intel's First Workaround

Part I, IDT Alignment:

A) Align the Interrupt Descriptor Table (IDT) such that it spans a 4-KB page boundary by placing the first entry starting 56 bytes from the end of the first 4-KB page. This places the first seven entries (0–6) on the first 4-KB page, and the remaining entries on the second page.

B) The page containing the first seven entries of the IDT must not have a mapping in the OS page tables. This will cause any of exceptions 0–6 to generate a page not present fault. A page fault prevents the bus lock condition and gives the OS complete control to process these exceptions as appropriate. Note that exception 6 is the invalid opcode exception, so with this scheme an OS has complete control of any program executing an invalid CMPXCHG8B instruction.

Part II, Page Fault Handler Modifications:

A) Recognize accesses to the first page of the IDT by testing the fault address in CR2. Page not present faults on other addresses can be processed normally.

B) For page not present faults on the first page of the IDT, the OS must recognize and dispatch the exception which caused the page not present fault. Before proceeding, test the fault address in CR2 to determine if it is in the address range corresponding to exceptions 0–6.

C) Calculate which exception caused the page not present fault from the fault address in CR2.

D) Depending on the operating system, certain privilege level checks may be required, along with adjustments to the interrupt stack.

E) Jump to the normal handler for the appropriate exception.

This is an ingenious solution to a horrible problem. Unfortunately, the solution is as bad as the problem. When the processor receives any of the first seven exceptions (Divide by Zero through Invalid Opcode), the processor generates a page fault instead of the appropriate exception. The page-fault handler gets mucked up with all kinds of code to check privilege levels and whether the fault was caused by another exception. If I had my druthers, I'd stay as far away from this solution as possible.

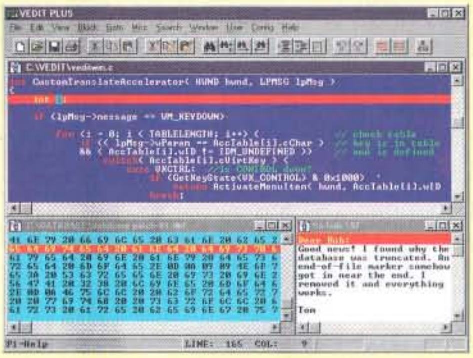
Intel's Second Workaround

Part I, IDT Page Access:

A) Mark the page containing the first seven entries (0–6) of the IDT as read only by

WOW! VEDIT can do that!

ASCII, EBCDIC, Hex, 1000+ Megs, Database, CD-ROM, Mainframe, Postscript, C, HTML, editing, translating, converting, and letters to mom, the new VEDIT PLUS 5.1 for Windows/DOS does it all.



VEDIT PLUS

"I love VEDIT. I replaced a difficult-to-maintain 3000 line text conversion program written in C with a 300 line VEDIT macro. Best of all, the VEDIT macro is faster."
S. Krishna, Quadrant, TX

Visit our Web site at www.vedit.com and download a fully functional demo.

Originally created in 1980 as the first programmer's editor, the new VEDIT PLUS 5.1 has evolved into a unique multi-function tool that effortlessly handles all of your routine, tricky and huge file editing needs.

Only VEDIT is fast enough for huge 100+ Megabyte files. It can perform thousands of search/replace operations in little more time than it takes to just copy the file.

Written mostly in assembly language, it's the most responsive program you'll ever use. Instant startup, instant everything.

Advanced features include multi-file, multi-window editing, template editing, syntax highlighting, search/replace with regular expressions and columnar blocks. Fully configurable keyboard and emulations.

VEDIT PLUS - Windows/DOS: $149. (Includes 32-bit Win 95/NT, 16-bit Win 3.1 and DOS versions.) 1-year free tech support. 30-day money-back guarantee. Attractive site license pricing. Also VEDIT: only $89.

Lightning fast multi-purpose tool

- Edit **absolutely any file** - text, data, or binary up to 2 Gigabytes.
- Edit in ASCII, Hexadecimal, Octal, EBCDIC or any combination of modes.
- Edit DOS, Unix and Mac text files, data files with fixed / variable length records.
- Translate between EBCDIC and ASCII, IBM PC and ANSI. Flexible sorting.
- Perform numerous search/replace on entire groups (even thousands) of files.
- Powerful C-like macro language for writing custom translators, filters, more.
- Exceptionally compact. Easy to install.

New VEDIT PLUS 5.1 features

- 32-bit version optimized for Win 95/NT.
- Attractive ANSI and OEM editing fonts.
- Convert newlines, smart detab/retab.
- Enhanced HTML and C editing.
- Improved word processing, searching, printing, "ctags", and much more.

VEDIT is a registered trademark of Greenview Data Inc.

Toll Free: 1-800-45-VEDIT (1-800-458-3348)
Telephone: (734) 996-1300, Fax: (734) 996-1308
E-Mail: sales@vedit.com, Web: www.vedit.com
Mail: P.O. Box 1586, Ann Arbor, MI 48106

Greenview Data

AD LINK 349

setting bit 1 of the page table entry to zero. Also set CR0.WP (bit 16) to 1. Now when the invalid opcode exception occurs on the locked CMPXCHG8B instruction, the processor will trigger a page fault since it does not have write access to the page containing entry 6 of the IDT. This page fault prevents the bus lock condition and gives the OS complete control to process the invalid operand exception as appropriate. Note that exception 6 is the invalid opcode exception, so with this scheme an OS has complete control of any program executing an invalid CMPXCHG8B instruction.

When the processor encounters the instruction F0 0F C7 C8, the bug occurs

B) Optional: If updates to entries 7-255 of the IDT occur during the course of normal operation, page faults should be avoided on writes to these IDT entries. These page faults can be avoided by aligning the IDT across a 4-KB page boundary such that the first seven entries (0–6) of the IDT are on the first read only page and the remaining entries are on a read/writeable page.

Part II, Page Fault Handler Modifications:
A) Modify the page fault handler to calculate which exception caused the page fault using the fault address in CR2. If the error code on the stack indicates the exception occurred from ring 0 and if the address corresponds to the invalid opcode exception, then pop the error code off the stack and jump to the invalid opcode exception handler. Otherwise continue with the normal page fault handler.

This workaround is really quite clever, in that it takes advantage of the bug as a means to provide a fix to the problem. When any of the first six exceptions occur, they are handled as they normally would. Divide by Zero through BOUND exceptions vector to their normal exception handlers without any intervening code in the page-fault handler. However, when the F00F bug occurs, the page-fault handler is invoked instead of the #UD handler. Why? CR0.WP=1 instructs the microprocessor to generate a page fault when

WINDOWS/WEB/DBMS/UNIX DEVELOPERS

great jobs & career info

- Get the latest job listings from **Start-ups** to **Fortune 500** companies e-mailed directly to you. It's easy & FREE.
- Stay informed of industry demands.
- Visit our website before April 30 for a chance to win a mountain bike.

HOT NEW SITE
www.developers1.net

Employers & Hiring Managers, is your job listed with DEVELOPERS1.NET? Visit our website or call us today!
408.469 3726

Visit our website today- you could **WIN A MOUNTAIN BIKE!**
www.developers1.net

D1013

developers1.net
A Community of Software Developers

The world's fastest SCM system

runs seamlessly on over 30 platforms:

BSDI	*Linux Alpha*	*NCR*	*Sequent*
BeOS	*Linux Intel*	*NeXT*	*Siemens*
Digital Unix	*Linux Sparc*	*OS/2*	*Solaris*
Free BSD	*Lynx*	*Open VMS*	*SunOS*
HPUX	*Machten*	*QNX*	*WindowsNT*
IBM AIX	*Macintosh*	*SCO*	*Windows95*

. . . And more!

(Maybe you should check us out.)

PERFORCE™

THE FAST SOFTWARE CONFIGURATION MANAGEMENT SYSTEM

info@perforce.com www.perforce.com
2411 Santa Clara Ave., Suite 40 Alameda, CA 94501 1-510-864-7400
© 1998 by Perforce Software, Inc.

AD LINK 376

Sequence	Address	Data	Mnemonic	Timestamp
428677	00060FF8	0008049A	(LOCKED MEM READ)	330 ns
	00060FFC	00008E00	(LOCKED MEM READ)	
428678	0001E8B8	BFF0FFFF	(LOCKED MEM READ)	170 ns
	0001E8BC	00009B01	(LOCKED MEM READ)	
428679	0001EE9C	00000008	(LOCKED MEM WRITE)	130 ns
428680	0001EE98	000003F2	(MEM WRITE)	60 ns

Example 2: *F00F bug on noncacheable page.*

Sequence	Address	Data	Mnemonic	Timestamp
429135	00060FF8	0008049A	(LOCKED MEM READ)	330 ns
	00060FFC	00008E00	(LOCKED MEM READ)	
429136	0001E8B8	BFF0FFFF	(LOCKED MEM READ)	170 ns
	0001E8BC	00009B01	(LOCKED MEM READ)	
429137	0001EE9C	00000008	(LOCKED MEM WRITE)	140 ns
429138	0001EE98	000003F2	(MEM WRITE)	50 ns

Example 3: *F00F bug on page write-through.*

Sequence	Address	Data	Mnemonic	Timestamp
426333	00060FF8	00080496	(LOCKED MEM READ)	60 ns
	00060FFC	00008E00	(LOCKED MEM READ)	
426334	0001E8B8	BFF0FFFF	(LOCKED MEM READ)	170 ns
	0001E8BC	00009B01	(LOCKED MEM READ)	
426335	0001EEA0	00010002	(LOCKED MEM WRITE	120 ns
426336	0001EE9C	00000008	(MEM WRITE)	60 ns
426337	0001EE98	000003EE	(MEM WRITE)	50 ns

Example 4: *F00F bug with cache disabled.*

an attempt is made by the supervisor to modify a memory page. The processor doesn't actually attempt to modify the Interrupt Descriptor Page (IDT page holding the #UD vector address) when the F00F opcode is encountered. But the bug actually makes the processor think it's modifying the IDT page with the #UD vector. The locked memory cycle somehow convinces the internal state of the Pentium to think that a write cycle is going to occur. Since the transition to the #UD handler is considered a supervisor task, the processor thinks it's going to write to this page. Thus when CR0.WP=1, a page fault occurs.

Even though this is a clever fix, there are two things I don't like about it:

- The fix requires a kludge to the page-fault handler code. The page-fault handler must contain code to detect this condition, and vector to the #UD handler.
- Setting CR0.WP=1 isn't a viable solution for everybody. Some operating systems may not be able to use this workaround.

If I were forced to choose between two of Intel's "blessed" solutions, I'd choose this one. However, because Intel set a precedent in documenting a solution that actually takes advantage of the bug behavior, this could give rise to much more elegant solutions that also take advantage of the bug behavior.

In fact, there are several alternative solutions to the problem, some quite simple.

Alternate Solution #1

Part I, IDT Alignment:
A) Align the Interrupt Descriptor Table (IDT) such that it spans a 4-KB page boundary by placing the first 56 bytes (IDT entries 0–6) from the end of the first 4-KB page. This places the first seven entries (0–6) on the first 4-KB page, and the remaining entries on the second page.

Part II, Page Table Modification:
A) Mark the first 4-KB page of the IDT as noncacheable by setting the appropriate Page Table Entry to Page Cache Disable (PTE.PCD=1).

All exceptions vector to their appropriate interrupt handler. The page fault handler doesn't need to be mucked up with any extra code. All of the exception handling code may remain unmodified. When the F00F bug occurs, the processor issues the two consecutive locked reads. However, the processor doesn't lock up because the page is noncacheable. Example 2 shows the logic analyzer trace of the microprocessor recovering from the F00F bug.

Alternate Solution #2

Part I, IDT Alignment:
A) Align the Interrupt Descriptor Table (IDT) such that it spans a 4-KB page boundary by placing the first 56 bytes (IDT entries 0–6) from the end of the first 4-KB page. This places the first seven entries (0–6) on the first 4-KB page, and the remaining entries on the second page.

Part II, Page Table Modification:
A) Mark the first 4-KB page of the IDT as noncacheable by setting the appropriate Page Table Entry to Page Write-Through (PTE.PWT=1). This is by far the best solution.

This solution maintains all of the benefits of having the page cacheable. However, because the page is considered write-through, the processor is tricked into recovering from the bus LOCK up condition. Example 3 shows the results of encountering the F00F bug when the page is cacheable, but marked as write-through.

Alternate Solution #3 (For DOS Users)

A) Turn off your cache. Enter your BIOS setup utility, and disable the processor cache. This prevents the F00F bug from locking up your computer.

This isn't really a viable solution for most people. Turning off the microprocessor cache can have a dramatically negative performance impact on your computer. Example 4 shows the results of the F00F bug with cache disabled.

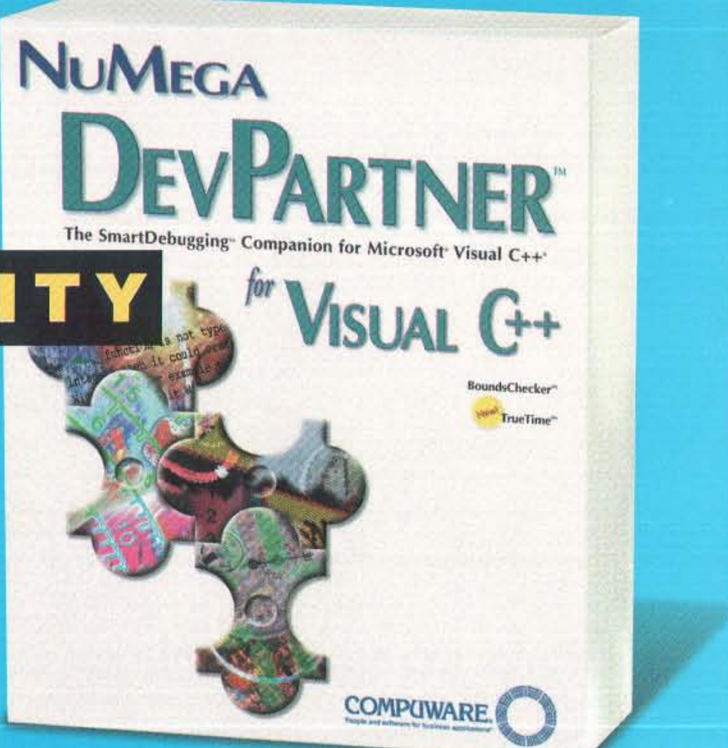
Conclusion

The ultimate solution to the F00F bug problem is obtained when disabling the cache—indicating that some interaction exists between the cache and the bug. Instead of taking advantage of the cache interaction, Intel's second solution takes advantage of interaction between the bug and page-fault mechanism. Now that Intel has set the precedent of using the bug behavior as a workaround, nobody should be concerned by the two more elegant solutions provided herein. My second alternate solution is by far the best. The exception handlers don't need to be mucked up with extra code, and the processor performance isn't impacted in the slightest. Note, however, that neither of these two alternate workarounds have been thoroughly tested in production code.

I've written two programs so that you can examine this bug and the various workarounds: F00FBUG.EXE, which demonstrates all five workarounds for the bug, and F00FBUG2.EXE, which demonstrates the most elegant workaround—Alternate Solution #2. The source code and executables for both programs are available electronically from both *DDJ* (see "Resource Center," page 3) and at ftp://ftp.x86.org/dloads/F00FBUG.ZIP.

DDJ

EXPAND YOUR TEAM'S PRODUCTIVITY WITHOUT EXPANDING YOUR TEAM.



NuMega DevPartner for Visual C++ Helps Developers Deliver Windows Applications Faster.

Compuware® NuMega™

Finding good experienced developers to crank out defect-free usable software is tough (not to mention expensive). But with NuMega DevPartner for Visual C++, you can turn up the productivity and capability of your existing team. Its suite of SmartDebugging tools for Microsoft Visual C++ accelerates development of Visual C++ components and applications for the enterprise and the Internet. Using DevPartner for Visual C++, individual developers can deliver high quality well-tuned applications and components almost as fast as whole teams can. That's because it automatically detects, diagnoses and facilitates resolution of software errors and performance problems. Get faster development and better code—all in one value-priced package—without adding developers to your team. Order NuMega DevPartner for Visual C+ today.

Microsoft Enterprise Development Partner

1-888-686-3403 www.numega.com

SmartDebugging™ for faster development VISUAL C++® VISUAL BASIC® JAVA™

Copyright © 1998 Compuware Corporation. All rights reserved. All trademarks and registered trademarks are the property of their respective owners.

30-DAY MONEY BACK GUARANTEE

We admit that on occasion we've even asked you to rip and replace what's inside your head.

But this is not one of those occasions. Instead we've got a plan to help you build the latest enterprise solutions using technologies you're already familiar with. Microsoft® Windows NT,® COM, and the Visual Studio™ development system can help you make it happen. It doesn't even matter if you have UNIX, NetWare or a legacy system that needs to be integrated. Hey, this whole distributed computing thing gets quite a bit easier when you use what you have, and more importantly, use what you know. To find out more about building new solutions with Microsoft Windows NT and Microsoft Visual Studio go to www.microsoft.com/msdn

© 1998 Microsoft Corporation. All rights reserved. Microsoft, Visual Studio, *Where do you want to go today?* and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners. Boris Karloff licensed by Karloff Enterprises. Represented by the Roger Richman Agency, Inc., Beverly Hills, CA. Frankenstein is a trademark and copyright of Universal City Studios, Inc. Licensed by Universal Studios Licensing, Inc. All rights reserved.

Microsoft
Where do you want to go today?®

Microsoft®
BackOffice™

Microsoft®
VISUAL Studio™
Visual Basic®
Visual C++®
Visual InterDev™
Visual J++™
Visual FoxPro®

Extending Windows CE 2.0 MFC Database Classes

Adding powerful search capabilities

John C. Schettino, Jr.

Windows CE programming is a lot like Win32 programming, since the CE API is a subset of Win32, with some CE-specific extensions. One of those extensions is an API for object databases. The object database API is reasonably rich, including indexed access to records in databases, with up to four indexes supported for each database. Indexes are only on single integer or string properties, in ascending or descending order. There is a *seek* function in the API that uses the current index for rapid searches. Like its bigger siblings, Windows CE has a version of the Microsoft Foundation Class (MFC) library, which I consider the highway— or at least the paved road— for developing CE applications. By comparison, using Win32/C is more like a gravel road. But because MFC in Version 1.0 of the CE SDK did not provide classes for dealing with databases, I hit the gravel road pretty often.

Just when I was about to break down and write my own wrapper classes, the CE SDK Version 2.0 beta was released. Thankfully, the SDK 2.0 adds several classes to MFC that wrap the database API and

data structures into a more usable form. While this API is sufficient to accomplish most database tasks, it still lacks a general-purpose searching class. Consequently, in this article, I'll present a set of CE database classes and subclasses of the new 2.0 MFC classes that provide an object-oriented wrapper to the basic database search API. The complete wrapper is available electronically; see "Resource Center," page 3.

What's New in SDK 2.0

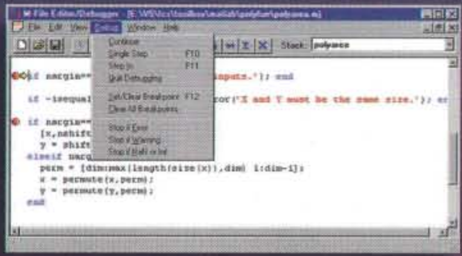
The main database class is *CCeDBDatabase*. Objects of this class encapsulate an open database, and support several operations. The operations I'm most interested in for searching are *SeekFirst()*, *SeekFirstEqual()*, *SeekNext()*, *SeekNextEqual()*, *SeekValueSmaller()*, and *SeekValueGreater()*. The *SeekFirstEqual()* and last two seek methods take a *CCeDBProp* object as their only parameter, and use the current index to locate the first record equal to, smaller, or larger than the value encapsulated within. *CCeDBProp* is a wrapper class for both record properties and index properties. It contains a property identifier and value. The *CCeDBRecord* class encapsulates a record in the database, and consists of a set of *CCeDBProp* objects. It provides a couple of methods for extracting a property from the record.

While there is a lot more to these classes than I've covered here, it's about all I need to extend the classes to add a more robust search capability.

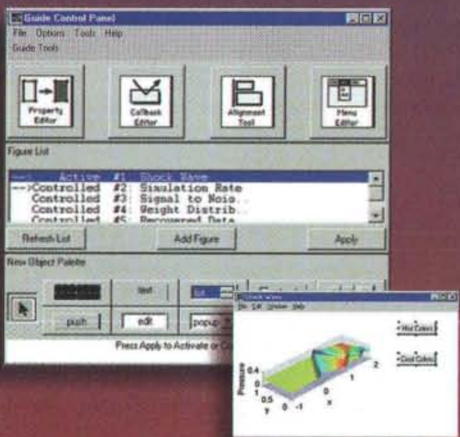
Objectives of the Search Classes

What I'd like to be able to do with MFC is encapsulate a query on an open database in an object, hand that object to a method of a database object, and get back an "iterater." I could then retrieve matching records by simply calling the *first()* and *next()* methods of iterater. The query would support the intrinsic queries supported by the database, that is less

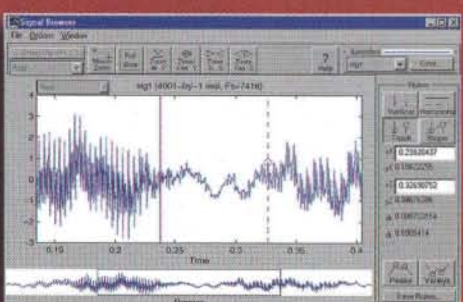
John is a developer of Windows CE, Newton, and Macintosh software. He is also a Senior Member of the Technical Staff at GTE Laboratories Inc. He can be contacted at schettino@writeme.com or at http://members.aol.com/pdcjohns/.



Application Development Tools
"We initially coded algorithms in MATLAB and then converted the MATLAB source to C or C++. To our surprise, the MATLAB code was faster in nearly all cases."
Jack Staub
Hughes Aircraft



Interactive GUI Design
"In one day, I wrote 875 lines of MATLAB which equates to 5,000 lines of C code. I had a functioning GUI in one day. You can't do that with C."
Kathleen Splaine
Risk International



Analysis and Visualization
"Anything from simple analysis to complex modeling and simulation can be done in a fraction of the time it would take to write your own code."
Gregory E. Chamitoff, Ph.D.
NASA, Johnson Space Center

We wrote exactly 698,794 lines of C code so that you don't have to.

More than 400,000 engineers and scientists use MATLAB to accelerate their technical programming. Here's why.

Faster programming

Today's most productive technical professionals have one thing in common – they use MATLAB instead of C or C++. Because, unlike a general purpose language, MATLAB is a complete, integrated analysis, visualization, modeling, and development environment specifically designed for technical computing. So development goes much faster and code is dramatically shorter.

MATLAB 5 COMPILER NOW AVAILABLE

More numerical power built in

At the heart of MATLAB is an easy to learn technical computing language with more than 500 functions built in. The MATLAB language includes flow control, multidimensional arrays, user-defined structures, ASCII and binary external data file access, and much more. And you'll save even more time with MATLAB Toolboxes, collections of highly optimized, discipline-specific functions written by world-class experts. You can even link in C, C++, and Fortran routines.

Less time coding means more time to think

Put simply, use MATLAB and it will take you far less time to develop finely tuned applications with revealing graphics, custom GUIs, and compact maintainable code. Now just imagine what you can do with the time you save.

See how MATLAB 5 can help you work faster. Visit our Web site for demos, examples, and updating information.

The MATH WORKS Inc.

MATLAB®
www.mathworks.com/ddjm
call 508-647-7000
e-mail info@mathworks.com

The MathWorks, Inc. 24 Prime Park Way, Natick, MA 01760 Fax 508-647-7001
Employment opportunities: http://www.mathworks.com/newjobs.html

The MathWorks is represented in the following countries: **Australia:** + 61-2-9922-6311 • **Benelux:** + 31(0)182-53-7644 • **Brazil:** + 55-11-816-3144 **Czech Republic:** 42-(0)2-68-44-174 • **France:** 33-141-14-67-14 • **Germany/Austria:** + 49-241-470750 • **India:** + 91-80-5-549338 **Israel:** + 972-3-561-5151 • **Italy:** + 39-11-240-80-00 • **Japan:** + 81-3-5978-5410 • **Korea:** + 82-2-556-1257 • **New Zealand:** + 64-7-839-9102 **Nordic Countries:** + 46-8-15-30-22 • **Poland:** + 48-126-17-33-48 • **Singapore/Malaysia:** + 65-842-4222 • **South Africa:** + 27-11-325-6238 **Spain/Portugal:** + 34(9)3-415-49-04 • **Switzerland:** + 41-31-954-2020 • **Taiwan:** 886-2-505-0525 • **United Kingdom/Ireland:** + 44-1223-423-200

© 1998 by The MathWorks, Inc. All rights reserved. MATLAB is a registered trademark of The MathWorks, Inc. Other product or brand names are trademarks or registered trademarks of their respective holders.

AD LINK 205

(continued from page 70)
than, greater than, or equal to tests. Unlike the basic index seek operations, I'd like to be able to use more than one property in the query, as well as supply multiple matching tests for a property. Finally, I'd like to lift the restriction on using nonindexed properties in a query. To top it all off, I'd like the query to be as optimal as possible, including making full use of the correct index to retrieve all the matching records while accessing as few records as possible in the database.

This sounds wonderful, but without a few simplifying assumptions I'd end up coding it for the rest of my life. In addition to the good things I've decided on, I'll make one huge concession—I'll assume that any query that contains multiple properties, or even multiple tests against a single property, is an implicit AND of all tests. Sure, I could support OR, but then I'd need a more complex representation of the query syntax as well as a query parser. I'll leave that as an exercise for the reader. Also not implemented but equally desirable are more general tests. It would be nice to be able to do regular expression matches or lots of other tests, and it's obvious exactly where those tests could be added, but I'm not doing them now.

Class Design

To implement extended searching, I'll first need to create the *CEQuery* class to encapsulate a query. That will require subclassing the *CCeDBProp* class so I can store test criteria with a property. Then I'll need to extend the existing *CCeDBDatabase* class to add the method to create an iterater used to search. Finally, I'll have to do the hard work of actually implementing the iterater class.

The *CEQuery* and *CCeDBQueryProp* Classes

A query object consists of a *CCeDBDatabase* object, a number of *CCeDBQueryProp* objects representing the properties to test, a test value (and the test operation), and a *CCeDBQueryProp* object representing the desired sort order (if any). Listing One (listings begin on page 110) presents the *CEQuery* class declaration.

The *addField()* method adds a new *CCeDBQueryProp* object to the query. It's expected that *CEQuery* deletes this object in its destructor. *CCeDBQueryProp* objects are stored in an array named *_fields* that is expanded as needed, so any number of tests can be performed in a query.

The *setSortField()* method takes a *CCeDBQueryProp* object (which is also deleted in the destructor) that specifies the desired sort order for the query. The *numFields()* method returns the current number of test fields in the query.

The *testOn()* method is called during searching. It determines which *CCeDBQueryProp* value (if any) to use to begin the query when querying the database. Each *CCeDBQueryProp* is tested against the supplied property ID, which is also saved in the *_index* member variable. If a matching *CCeDBQueryProp* uses an Equal test, then that object is selected to begin the query. If it matches and uses a Less Than (on a property with a descending index) or Greater Than (on a property with an ascending index) then it is used if none are found with an Equal test. At this time if a range search is detected (both a Greater Than and Less Than test on the indexed property, for example) then the *_range* member variable is set to True. The goal of this method is to start the search off using a query value against the selected index that eliminates as many nonmatching records as possible. It also determines (by setting *_range*) if it is possible to stop the query after a record fails a test on the indexed property.

The *match()* method is called during searching. It tests every property in the supplied record against the *CCeDBQueryProp* objects in the query. If a range test is being done, the index property is tested first to see if it falls within the range. If it does not, then False is returned, but also an EOF flag is set indicating that the search is complete. The EOF flag is not set the first time the range test fails, to work around a bug in the MFC *SeekValueSmaller()* method. When used with a descending index, it actually returns the record larger than the supplied value, so the first time, the range test will fail. It's not too expensive to do a couple of extra tests just to be sure, and it works around this bug just fine. The *match()* method calls *matchField()* to do the actual test for each property. Assuming the range test succeeds or is not performed, then each property in the record is tested against its corresponding *CCeDBQueryProp* objects using *matchField()*. When doing these tests the index field tests are skipped if range testing was previously done. If all tests pass, *match()* returns True.

The *matchField()* method tests the supplied property from a potentially matching record against the specified test object, and returns True or False. It consists of a large case statement that does the appropriate comparison based on the data type and test operation for a property.

The *CCeDBQueryProp* class is a subclass of *CCeDBProp* (see Listing Two for its declaration). It adds an enumerated type that describes the three types of test possible on a property, as well as a member variable to hold the specific test for the property. To create a query, I allocate a new *CCeDBQueryProp* object for each desired test, set its property ID, type, value, and operation, and then add it to the query object using *addField()*.

CCeFindDBDatabase

I'd like to be able to hand my query object to a method of a *CCeDBDatabase* object to begin a query. That's accomplished by subclassing *CCeDBDatabase* and adding a *Find()* method; see Listing Three.

The *Find()* method accepts a query object and allocates a new *CCeFindDBDatabaseIterater* object. This object should be deleted when I'm done with a query. To create a new iterater, I call the *FindKey()* method to determine which key I should use, and then I pass that key and the *CEQuery* object to the *CCeFindDBDatabaseIterater* constructor.

The *TestFind()* method is useful to determine if a particular query will use an index or require a sequential search. It also calls *FindKey()*, and if no key is found it returns False. This might be handy if I want to warn the user that a query might take some time, since it will be a sequential search.

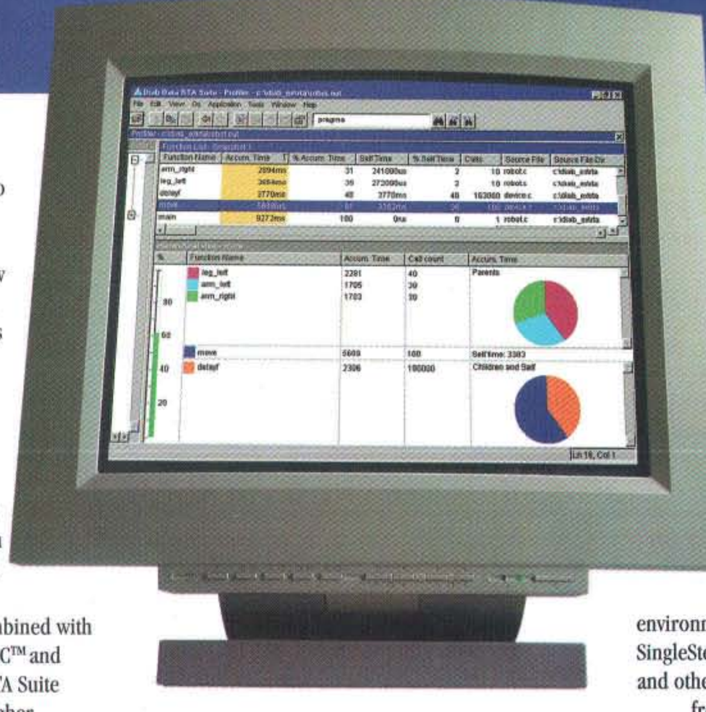
The *FindKey()* method examines the *CEQuery* object against the current indexes defined for the database using the *IsIndexed()* method. It determines which index (if any) to use when querying the database. If the query includes a sort then that index must be used. If not, then the *CCeDBQueryProp* objects that use properties that are indexed and weighted to determine the best index to use. If any of them are Equal tests, then that index is heavily weighted. If any are Less Than (on a property with a descending index) or Greater Than (on a property with an ascending index) then they are weighted at half of properties using Equal. The highest weighted index is then chosen. This ends up favoring the index that will require the fewest number of tests to complete the query.

The *IsIndexed()* method compares the supplied property ID against all property IDs in the database that have an index. If the property ID is indexed, it returns True, and also returns the sort order for the index.

CCeFindDBDatabaseIterater

When I query a database, I want a single object to encapsulate everything having to do with the search in progress. The *CCeFindDBDatabaseIterater* class holds everything needed to perform a search, including copies of the current

SEE WHAT YOU HAVE BEEN MISSING



End The Search!

Looking for development tools to improve program performance, reliability, and memory usage? Look no further! Diab Data's new RTA Suite represents a quantum leap forward in the quest for this exciting new class of tools.

The RTA Suite

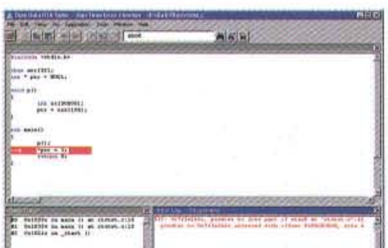
Diab Data's RTA (Run-Time Analysis) Suite allows you to capture and analyze your code's run-time behavior, providing you with the critical run-time insight you need to maximize program performance and reliability. Combined with Diab Data's industry-leading D-CC™ and D-C++™ compiler suites, the RTA Suite enables you to develop faster, higher quality code in less time.

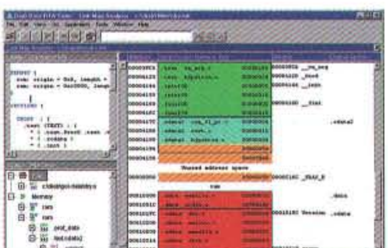
...With The RTA Suite

Powerful Run-Time Analysis Tools

A Visual Interactive Profiler for a revealing view of run-time program behavior, including hierarchical "top-down" analysis of parent/children function pairs.

A Run-Time Error Checker for detecting memory leaks, hard-to-find pointer errors, stack overflows, and many other conditions associated with invalid memory references at run-time.





A Visual Link Map Analyzer for interactive "drag-and-drop" optimization of memory maps.

For Every Developer

The simplicity of the RTA Suite's graphical user interface makes it useful for every developer on your team. Now every developer can benefit from application specific optimizations and the ability to enhance code quality before Q.A. What's more, the RTA Suite is integrated with leading development environments such as ISI's pRISM+™, SDS' SingleStep™, Enea OSE Systems' Illuminator™, and others, so you can access the RTA Suite from familiar tools.

Diab Data: The Expert's Choice

Diab Data is the compiler leader for embedded PowerPC and Motorola RISC applications. So when you are ready to take your PowerPC, 68K/CPU32, ColdFire, or M•CORE project to the next level, don't fumble around in the dark. Call on the experts.

And see what you have been missing with the RTA Suite!

MOTOROLA EMBEDDED PLATINUM DEVELOPER

DIAB DATA
Defining Compiler Performance

Diab Data, Inc., Tel: 650.571.1700, Fax: 650.571.9068. Diab Data GmbH, Tel: +49 (0)89.9393.1191, Fax: +49 (0)89.930.5184

© 1998, Diab Data, Inc. All Rights Reserved. All trademarks used are the property of their respective holders.

Browse Us www.ddi.com Email Us info@ddi.com

(continued from page 72)
query object and the database object. Listing Four is its declaration.

The constructor stores copies of the *CEQuery* (query) and *CCeDBDatabase* (database) objects in the *_myQuery* and *_myDb* member variables. It then opens the database object in *_myDB* using the selected index. It sets *_start* to True, *_eof* to False, and zeros out the *_hits* and *_tests* member variables. It sets the *_primaryQuery* member variable to the query property to begin the search by calling the *testOn()* method of the *CEQuery* object. The iterater is now ready to retrieve matching records.

The *first()* method returns the first matching record for the query. It can be used to restart a search in progress, or just to begin a search. It just sets *_start* to True, clears the search statistics variables, and calls the *next()* method.

The *next()* method finds the first (if *_start* is True) or the next matching record in the database. If _start is True, it begins the query by calling *SeekFirst()* if there is no appropriate test on an indexed field. If there is an Equal test in *_primaryQuery* it calls *SeekFirstEqual()*. If there is a Greater or Less Than test in *_primaryQuery* it calls *SeekValueGreater()* or *SeekValueSmaller()*. If *_start* is False, then it calls *SeekNextEqual()* if there is an Equal test in *_primaryQuery*, or *SeekNext()* otherwise. In other words, it tries to find the first (or next) value using the index and the test value supplied in the query object. If the seek operation found a record, it's still not ready to return. Instead, the method enters a *while* loop calling *match()* with the record. If *match()* returns False but does not indicate that EOF has been reached, it gets the next record (again using *SeekNextEqual()*, or *SeekNext()*) and tries again. This continues until a record matches or EOF is detected. It then return the matching record or NULL if EOF was reached.

The *stats()* method returns the number of records tested and the number of matching records for the query. It can be called at any time, even during a query, to find out how many records have been examined and matched by the query.

Example Application

To test the classes, I wrote a simple MFC application that creates a database with 1000 records containing random string, long, and date (FILETIME) properties and then allows searches on it. The string property has an ascending index, and the long property has a descending index. The application has a simple form (see Figure 1) that accepts one or two strings, longs, and dates with search criteria, as well as a desired sort of none, string, or integer.

Selecting *Initialize* creates a new test database with 1000 records. The initialize routine uses the CE2.0 database classes to create a database with indexes on a string and integer field, and populate it with records containing random data; see Listing Five.

The database is created and if the *Create()* method fails, the existing database is deleted and then recreated. It's created with the two indexes (on property 1— an ascending sort on strings, and property 2— a descending sort on longs) I use to test the query classes. Finally, 1000 records are added with random strings, longs, and dates that fall between 1/1/1900 and 12/28/2100.

To perform a search, users fill in the desired criteria and select *Search*. The *OnSearch()* method extracts the current criteria, builds a *CEQuery* object, opens the database using a *CCeFindDBDatabase* object, and then uses the *Find()* method to get an iterater for the results. Listing Six is the portion of the search routine that creates one of the query properties in the query object and then performs the query.

For each query operation and test value in the form, it tests to see if the user entered a value. If the user did enter a value, it extracts the desired test and value from the dialog and creates a *CCeDBQueryProp* object. It then adds this object to the *CEQuery* object using *addField()*. Once all the properties are added, it opens the database and calls *Find()* to get the iterater. A simple *for/next* loop retrieves the matching records from the iterater. It uses the MFC TRACE() macro to display the properties of the matching records and search statistics in the debugger output window.

To test the classes, I performed a series of queries. The tests consist of doing single and then range searches on the string, long, and date fields, and then equality tests on each. Finally, a query using all three properties is done. For all these tests I tried selecting no sort order, a sort order that did not correspond to the query properties, and finally one that did. The results are encouraging. In cases where I did not select an index, but queried using the string (Greater Than, Equal tests) or integer (Less Than, Equal tests) properties the correct index was selected and the search completed without examining all records. In the case of range tests the fewest possible records were examined. Adding tests on the date

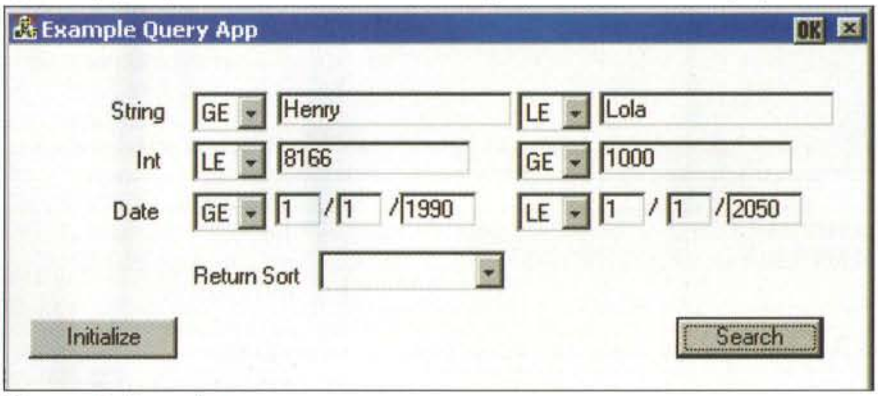


Figure 1: *Query form.*

```
Results:
        Hpynxotoa     4685    Wednesday, February 07, 2046
        Iuquajtn      8153    Wednesday, February 02, 1994
        Izpbopxe      6255    Saturday, August 17, 2024
        Jdhusu        7065    Wednesday, December 13, 2023
        Jkswfpq       6956    Saturday, April 07, 2012
        Jxgxxn        4137    Thursday, September 23, 2004
        Jyuhxve       5032    Thursday, July 14, 1994
        Jzkooz        2402    Tuesday, November 17, 2043
        Keqvdo        8098    Tuesday, September 16, 2025
        Kzggvswxua    4594    Friday, December 12, 2003
        Lgtarizsq     7428    Friday, September 18, 2026
        Lnrrxhmh      7922    Tuesday, October 21, 2031
Stats:
        matches: 12
        misses: 158
        Total: 170

Range search was used
```

Figure 2: *Output from a query.*

property properly excluded nonmatching records. If I did a test using the long property but selected the string sort order, records were returned in the correct order but an exhaustive search was (correctly) performed. In other words, the classes meet all the desired objectives. Figure 2 is output from a query including range tests in all three properties.

Conclusion

The classes I've presented here add a powerful search capability to the basic database classes found in Windows CE 2.0. They're

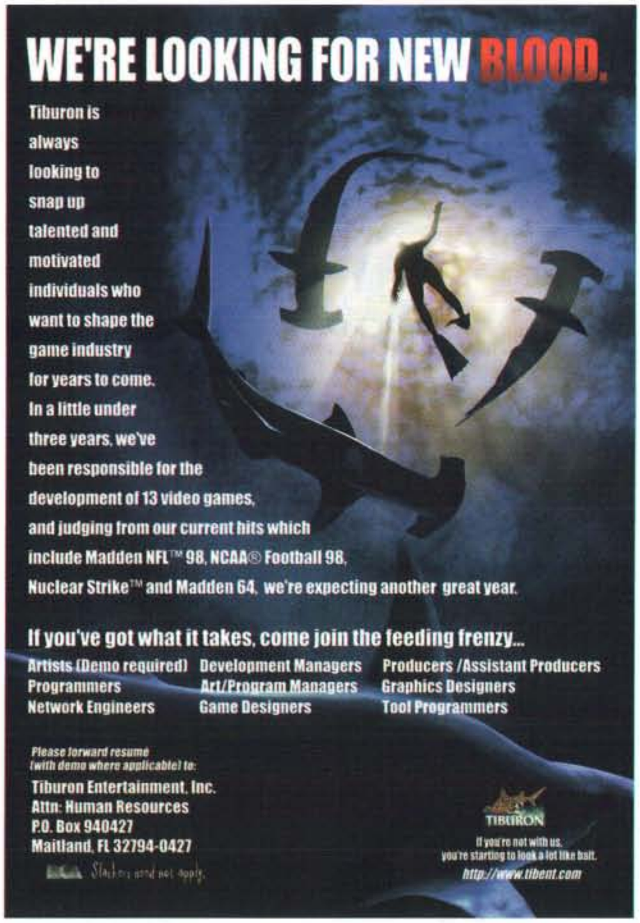
> *The SDK 2.0 adds several classes to MFC that wrap the database API*

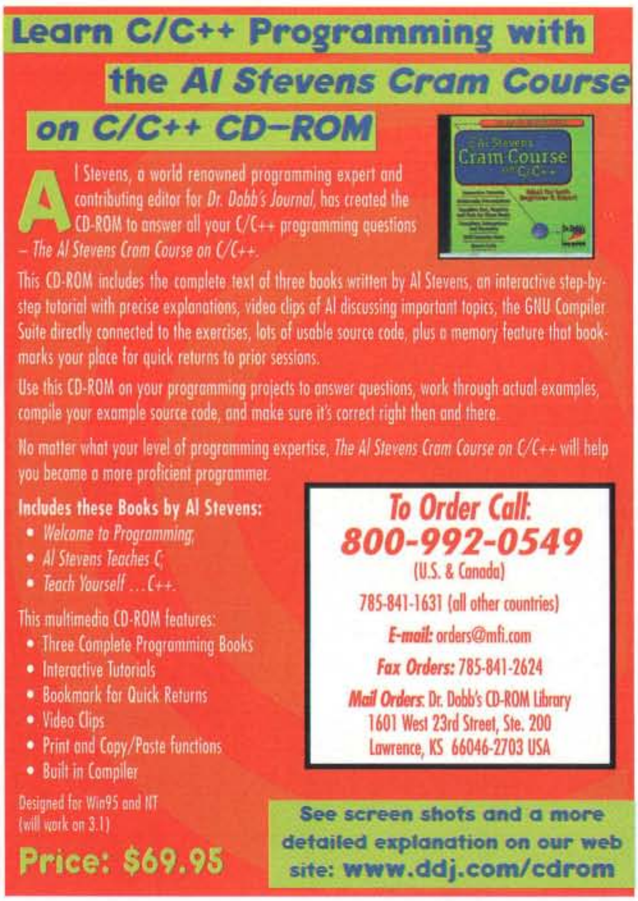
easy to use, provide fairly optimized searches, and allow for testing on every property in a database—not just strings and integers as is supported by the base class. The memory overhead of using the classes is reasonably low—a typical query object and iterater require only one to two KB of memory over the base classes.

There are some obvious enhancements that are possible: I could implement a class that would allow for Boolean operations for combining query properties (AND, OR, NOT) rather than the implicit AND I use now. That would require a more complex *match()* method. I could add more test operations including substring and regular expression matching. That would expand the *matchField()* method. There is one additional query optimization possible as well: The case where the query specifies a Less Than (or Greater Than) test on an index that is ascending (or descending) could use the index, and terminate the query as soon as a key value fails. This is basically half the range test I've already implemented, and it would be easy to add to the *testOn()* and *match()* methods. Even without these enhancements, the query classes are a powerful addition to my programming toolchest.

DDJ

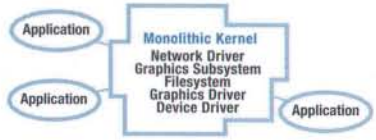
(Listings begin on page 110.)





Joe reboots his PC every day.

That's a fact.

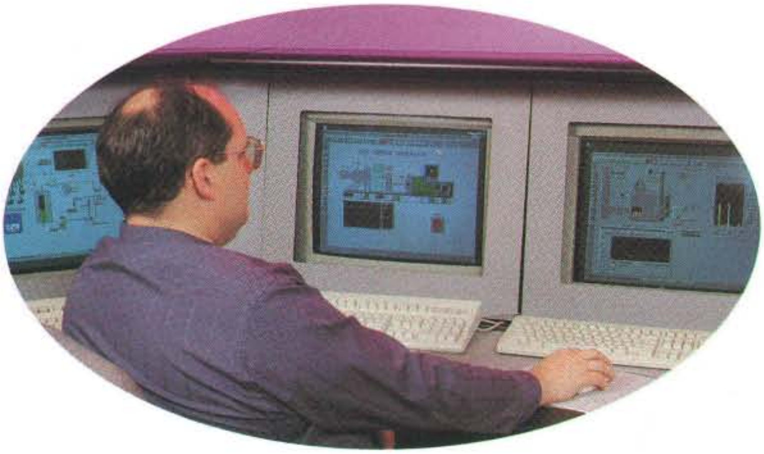


Conventional OS Architecture

*The monolithic OS on Joe's machine clumps all OS components into a single address space. One subtle programming error in just one driver, and **whoomp!**, Joe has to reboot – again.*

Dave hasn't since 1994.

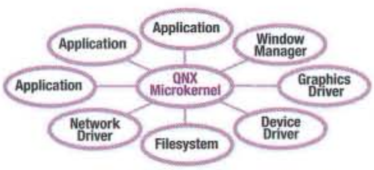
That's a fact too.



Four years ago, Dave Cawlfield at Olin Chemicals replaced expensive PLCs with OMNX Open Control Software and the QNX Realtime OS. "Since then," says Dave, "we've upgraded the control system regularly with new hardware and software – including parts of the OS itself. But not once have we had to reboot."

white paper

For a handy 12-point checklist on OS reliability, download Dave's paper, Which OS for PC-based Control?, *at www.omnx.com/productinfo/technical_papers.htm.*



QNX® Microkernel Architecture

*The QNX OS on Dave's machine runs **every** OS component in its own MMU-protected address space. So if a driver – or virtually anything else – fails, the rest of the system stays up.*

- Deterministic realtime performance (1.95 µsec per context switch on a Pentium 133)
- Full MMU support for *all* processes
- Small memory footprint
- Fault-tolerant networking
- Inherent distributed processing
- Internet and mobile SDKs
- Embedded GUI & Browser
- POSIX certified
- Embedded OEM pricing

Build Reliable Embedded Systems with QNX

Most operating systems work fine – until they hit a minor software glitch. Or until you perform a simple maintenance task, like changing an input device. Then, like it or not, the OS goes down – and your application with it.

With QNX, on the other hand, your system can recover from software faults, even in drivers and other critical programs. What's more, you can hot-swap peripherals. Start and stop filesystems and network services. Change I/O drivers. Add or remove network nodes. Even access the OS after a hard disk failure. **All without a reboot.**

And here's the kicker: QNX scales seamlessly from handheld consumer appliances to continent-spanning telephony networks. So you can use one OS to keep all your solutions running 24 hours a day, 7 days a week – nonstop. **And that's a fact.**

www.qnx.com *(don't miss our demo!)*
call **800 676-0566** *ext. 1364*

QNX®

The Leading Realtime OS for PCs

QNX Software Systems Ltd., Voice: +1 613 591-0931 Fax: +1 613 591-3579 Email: info@qnx.com
QNX Europe: Voice: +44 (0)1223 364503 or +1 613 591-0931 Fax: +44 (0)1223 364504 Email: QNXeurope@qnx.com
© QNX Software Systems Ltd. 1998. QNX is a registered trademark of QNX Software Systems Ltd. All other trademarks belong to their respective owners.

AD LINK 134

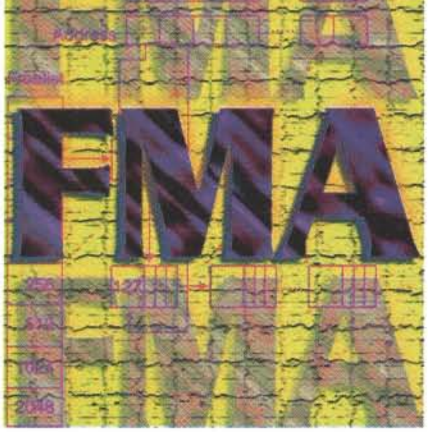
Fast Memory Allocation

Power-of-two for the PowerPC 403

H. Thomas Richter

Power-of-two memory allocators waste a substantial amount of memory—up to 50 percent in the worst case. In embedded-controller projects with execution-time constraints, however, it's okay to sacrifice memory for speed. The power-of-two Fast Memory Allocator (FMA) I present here is used in just such a project.

I developed the FMA as part of a hardware/software project that controlled digital and analog I/O lines between an IBM PowerPC 403-based coprocessor card and machines. (The PowerPC 403 is a family of 32-bit general-purpose embedded processors with data and instruction cache, MMU, DMA, fast interrupt latency, and low-power consumption; for details, see http://www.chips.ibm.com/products/embedded/.) We used the COP403 coprocessor card to run OS Open, an IBM internal POSIX-compliant real-time operating system. (Object code for OS Open is freely available at http://www.chips.ibm.com/products/ppc/documents/.) Attached to the coprocessor card was an InterBus-S field bus to read and write digital and analog I/O lines. The coprocessor card communicates via TCP/IP over an ISA bus with the Windows-NT host. Machine control programs were developed on Windows NT (using tools from a third party, which ported its tools to the coprocessor card and OS Open) and downloaded to the coprocessor card for execution. We were able to run complete machines, such as the FESTO MPS, with this card and software. (The FESTO MPS, manufactured by FESTO Inc., is a machine that uses analog and digital I/O for manufacturing metal parts.)



Thomas, an engineer for IBM Entwicklung GmbH, can be contacted at richter@vnet.ibm.com.

The applications that we developed needed:

- A fast, dynamic memory-allocation scheme to cope with burst requests and releases of equal-sized memory blocks.
- Many processes to share C++ objects with virtual functions. (Classes containing virtual functions maintain a virtual function table that had to reside on the same address location for all processes.)

To meet with these requirements, I tweaked the McKusic-Karels algorithm (also known as the "BSD 4.4 allocator") to optimize memory management for blocks fitting on a single page. (Requests spawning multiple pages are rounded up to the next page boundary. I don't currently support them, but support can easily be added using bitmaps.) Every memory request lower than or equal to the underlying operating system's page size is rounded up to the next integer y with $y=2^x$, $x \geq 4$, $y \leq$ *page size of OS*. This policy is used, for example, in the buddy-storage allocator (described by Donald Knuth in *The Art of Computer Programming, Volume 1)*, UNIX BSD 4.4 kernel heap system, and AIX operating system.

The disadvantage of the BSD and AIX implementations, however, is that the

Are you an embedded programmer who likes everything at your fingertips?

With Phar Lap's TNT Embedded ToolSuite®, you get the convenience of a winning combination for building embedded and realtime software. Use our RTOS with Microsoft® Visual C++® 5.0 and the Developer Studio IDE and get a total embedded development tools solution! Edit it. Build it. Assemble it. Link it. Cross-debug it. Check on-line help. And do it all from within a single, familiar, time-tested IDE.

With support for 32-bit x86 hardware, standard networking/Internet protocols, and Win32® development tools, the Realtime ETS Kernel leverages existing technology to enhance your productivity.

Visual C++ Comes Optimized for Embedded Systems

Visual C++ has all the right stuff for embedded development: an inline assembler, I/O port access functions, structured exception handling, and re-entrant ROMable runtime libraries. And its optimizing code generator is second to none when it comes to creating fast, tight code.

Version 9.0 of the TNT Embedded ToolSuite lets you do what no other package does—build realtime software with Visual C++ and Developer Studio, the most widely used compiler-IDE package available. Everything you need at your fingertips. Everything just mouse clicks away on the IDE toolbar.

The Only Hard Realtime Kernel that Works with Visual C++ 5.0

Phar Lap's Embedded StudioExpress™ add-in is the bridge between our Realtime ETS™ Kernel, Visual C++ and Developer Studio. It lets you use the integrated Developer Studio debugger as a cross-debugger with full hardware and kernel awareness. It's all just an icon click away on the Developer Studio toolbar.

Phar Lap's support of Visual C++ and Developer Studio gives you a total embedded development tools solution that is Simply On Target!™

Get Web-ready with the ToolSuite! See our MicroWeb Server technology for embedded systems in action at the "World's Smallest Web Server": **http://smallest.pharlap.com.**

AD LINK 99

Phar Lap and TNT Embedded ToolSuite are registered trademarks, and ETS and Embedded StudioExpress are trademarks of Phar Lap Software, Inc. Other product and company names are trademarks or registered trademarks of their respective holders.

The 32-bit x86 Experts

Phar Lap Software, Inc.
60 Aberdeen Avenue
Cambridge, MA 02138
Tel: (617) 661-1510
Fax: (617) 876-2972
http://www.pharlap.com/ddj
Email: info@pharlap.com

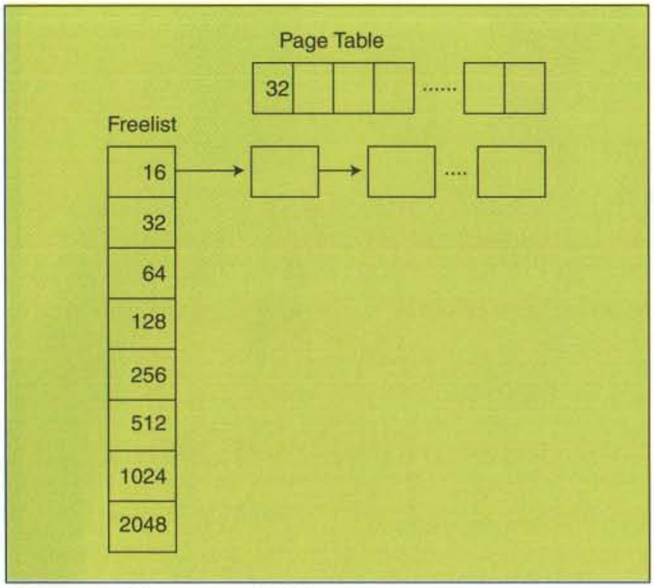


***Figure 1:** Allocation of 32 bytes.*

$$\text{idx} = \frac{\text{pointer–base address of memory region}}{\text{page size}}$$

***Figure 2:** Dividing by page size.*

$$0 = \frac{\text{0x30000FE0} - \text{0x30000000}}{4096}$$

***Figure 3:** Subtracting address from base address of the memory region.*

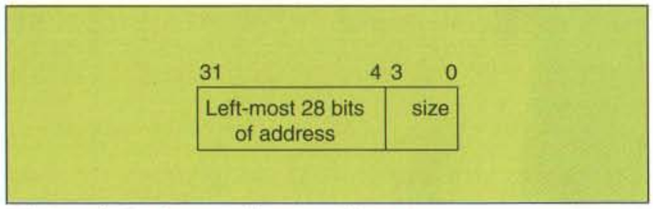


***Figure 4:** Bit layout of page-table entry.*

(continued from page 78)

buddy-storage allocator is time intensive, making it less suitable for real-time applications. Also, the UNIX BSD 4.4 memory allocator does not unite returned memory blocks, which leads to fragmentation.

The FMA algorithm presented here overcomes these disadvantages. Unused and returned memory chunks are united when possible, thus avoiding fragmentation. This is accomplished with very little overhead, so it is suitable for real-time applications. The complete source code for the FMA (for AIX) is available electronically; see "Resource Center," page 3.

The Algorithm

The FMA and McKusic-Karel algorithms use several structures:

- A continuous region of memory. The region is divided into pages and starts on a page boundary, which is four KB. Memory requests from applications, which must fit on a single page for the FMA to be handled, are rounded up to the lowest power-of-two number (16 bytes minimum). Thus, a four-KB page can be divided into chunks of 16, 32, 64, 128, 256, 512, 1024, or 2048 bytes without waste. Full pages are also possible.
- A page table. It has as many entries as there are pages in the memory region.
- The freelist. For each allocatable size possible, there is an entry in the freelist. It chains together blocks of the same size. The freelist is an array of eight elements.
- Pointers and counters for internal housekeeping. Last-used pointers indicate the last-used page in the region; anchor pointers indicate a released page in the region. And semaphores are employed to synchronize shared-access and maintain shared-memory descriptors.

When an application starts, it creates a memory region via a call. The application specifies the size of the region (if it has to be created) and whether it is sharable or private.

Memory is requested and released with subroutine calls identical to the *malloc* and *free* interface. The size of memory needed is specified in requests, and a pointer to a block of at least that size is returned. This address is used when the block is returned to the system.

McKusic-Karels

In the McKusic-Karels allocator algorithm, memory requests are satisfied from the corresponding freelist entry after the correct size has been determined. Each entry in the freelist is a pointer to a chain of free memory blocks of the same size. A null pointer indicates an empty list. If a request hits an empty list, a new page is obtained from the memory region. The page is divided into blocks that form a single linked list with the freelist entry as an anchor. The page-table entry for each page contains the size of the memory blocks the page was divided into. Figure 1 illustrates the situation after the first memory request of 32 bytes. The start address of the block is returned to the application.

When the block is later released, the base address of the memory region is subtracted from the current address, and the result is divided by the page size. The result is the index to the page table that contains the size of the memory block when it was originally allocated; see Figure 2. The block is then inserted into the corresponding freelist. Since the page size is usually a power-of-two number, the division can be replaced by a right-shift operation.

Assume the memory region was allocated at address 0x30000000. With a page size of four KB, this first page is mapped at address 0x30000000, the second at address 0x30001000, and so on. Further assume that the application requested a 32-byte block and received a pointer to address 0x30000FE0. If this pointer is now returned to the memory system, the address is subtracted from the base address of the memory region; see Figure 3. The page-table entry for page zero has a value of 32, so the block is inserted in the freelist array at element number one.

With many memory allocations and frees, each chain contains pieces from different pages. Memory is never united to form bigger blocks. A burst of requests for 16-bytes memory blocks renders a large area of the memory region useless for bigger requests even after all blocks have been returned.

FMA

The FMA stores additional information in the page-table entries and unallocated memory blocks:

- Three bits are used to identify the block size into which a page has been divided. The bits represent the index into the freelist array and indirectly reveal the block size of each element in the list.

- With the minimum block size set to 16 bytes per request, all blocks are aligned on a 16-byte boundary (or multiples thereof). The last four bits of any block's start address are zeros, and can be used for other purposes; see Figure 4. Each page-table entry is 32 bits wide, the same size for a pointer on a 32-bit processor. The *size* field of a page-table entry contains the index to the freelist array. With eight different sizes possible, four bits are sufficient. The address part contains the high-order 28 bits of the entry block start address (see next item). To form usable addresses, it must be multiplied by 16.
- Returned memory blocks can be used to store additional internal information. When a page is divided into blocks of a particular size, the corresponding freelist is empty. The first block of a page, called the "entry" block, contains the number of remaining blocks in the chain (including itself, a pointer to its successor, and two pointers for a double-linked list). The pointers for the double-linked list are used to chain entry blocks of the same size. Entry blocks are referenced from the freelist and from the page-table entry.

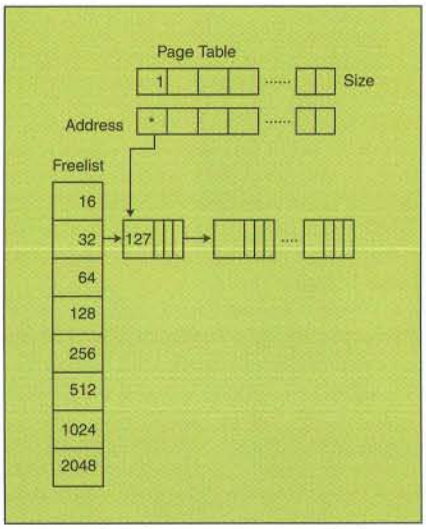


Figure 5: *Splitting page into 32-byte blocks.*

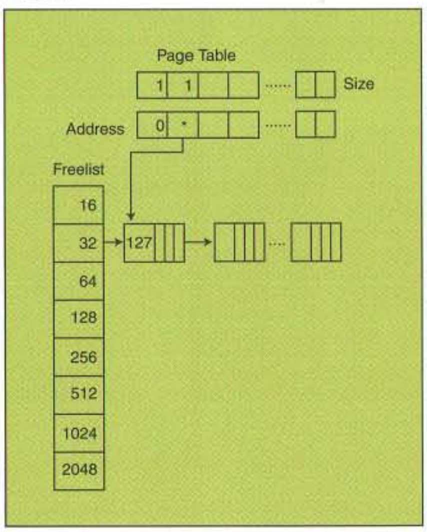


Figure 6: *Splitting second page into 32-byte blocks.*

Now assume the application requests a 32-byte block. The system allocates the first page and divides it into 128 blocks. The first 127 blocks are chained together to form a single-linked list. The entry block contains the number of blocks on this chain. The address of the entry block is stored in the freelist array at element one. The high-order 28 bits

$$\text{block count}=\frac{\text{page size}}{\text{block size}}$$

Figure 7: *Dividing page size by block size.*

of the entry block's address are also stored in the page-table entry, as is the index into the freelist array. The last block on the page is returned to the application. Figure 5 shows the state of the system at this point.

If another 32-byte block is requested, the system finds a valid pointer on the freelist array at element one, and returns the block following the entry block. The

Professional Development Tools for x86 Embedded Systems

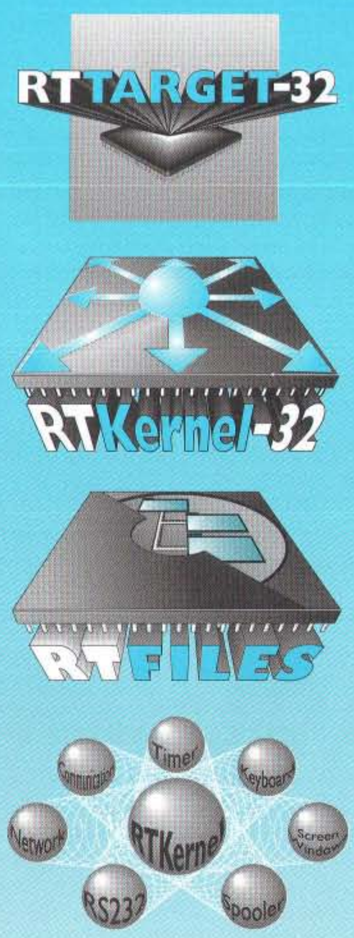
RTTarget-32
Cross development system for 32-bit embedded systems.
Boot code, locator, cross debugger.
For Borland C/C++, Microsoft C/C++, Watcom C/C++, and Delphi.
Supports Intel 386 and higher, as little as 16k RAM/ROM.
Windows NT compatible API!

RTKernel-32
High-performance real-time multitasking kernel for RTTarget-32.
For Borland C/C++, Microsoft C/C++, and Watcom C/C++.
Supports Intel 386 and higher.
Windows NT compatible API!

RTFiles
Embedded FAT12/16 file system for 16 or 32 bit embedded systems.
For Borland C/C++, Microsoft C/C++, and Watcom C/C++.
Supports Intel x86.
WinNT / DOS compatible API!

RTKernel
High-performance real-time multitasking kernel for DOS and 16-bit embedded systems.
For Borland C/C++, Microsoft C/C++, and Borland Pascal.
Runs under DOS and Paradigm Tools!

No run-time royalties!
Evaluation Kit available!



North America: On Time • 88 Christian Avenue • Setauket, NY 11733 • USA
Phone (516) 689-6654 • Fax (516) 689-1172 • email info@on-time.com
International: On Time Informatik GmbH • Hofweg 49 • 22085 Hamburg • Germany
Phone +49-40-2279405 • Fax +49-40-2279263 • email info@on-time.de

http://www.on-time.com

On Time
REAL-TIME AND SYSTEM SOFTWARE

block count in the entry block is decremented to reflect this.

If the block count drops to zero, the entry block itself is returned to the application. In this case, the freelist array element and the address part of the corresponding page-table entry must be updated. They will both be set to zero. The index of the page-table entry is calculated with the formula Figure 2.

If the application requests further 32-byte blocks now, a new page is allocated and the same scenario repeats. The freelist entry points to an entry block; this time, however, it is located on a different page (see Figure 6).

Release of Memory

If a memory block is released from the application, the page-table index is calculated using the formula in Figure 2. The address part of the page-table entry can behave in two ways:

- If it is not zero, it contains the high-order 28 address bits of an entry block for this page. The new block is appended after the entry block and the block count is incremented. If all blocks of a page have been released, the page can be reused.
- If it is zero, the first block of a page in use has been returned (this can be any block from that page). It will be the new entry block for that page. The page-table entry for this page refers to that entry block, and the block count is set to one. If no freelist entry for this block size exists, then the block is the only one available and the freelist entry refers to it.

If the freelist entry is already pointing to an entry block, the new entry block is the new head of the double-linked list.

Assume the application returns its first requested block (which resides on the first page). The page-table entry for this page indicates a block size of 32 bytes with zero for the address part, making this block the entry block. The freelist entry for 32-byte blocks is valid, so the entry blocks must be chained into a double-linked list, as in Figure 8. If another block from the first page is released, the address part of the page-table entry points to a valid entry block, and the newly released block is simply appended. The block count in the entry block is incremented.

Reuse of Pages

If all the blocks residing on a single page have been returned, the page can be reused (there is no need to unite adjacent pages since memory requests spanning multiple pages are not currently supported). All blocks have been returned when the block count in the entry block is the same as the page size divided by the block size; see Figure 7.

Since page size and block size can be expressed as power-of-two integers, the division can be replaced by subtraction. If the page is not used anymore:

- The entry block must be removed from the double-linked list of entry blocks. If the freelist element refers to this entry block, it is updated to point to a successor (or is set to null if no more entry blocks are on the list).
- The corresponding page-table entry is no longer referred to and can be reused to form a single-linked list of free pages.

The FMA allocation stores additional information in the page-table entries

The system maintains an anchor to the first free element in the page-table entry (named *anchor*), and a pointer to the last page allocated from the memory region (called *lastused*); see Figure 7. *lastused* points to the last page allocated in the region. If a new page is requested and *anchor* is null, *lastused* is incremented to the next page boundary until the end of the region is reached. If *anchor* is not null, it points to the page-table entry of a reusable page. The page-table entry contains a successor (if any), or –1 (which terminates the linked list).

For example, assume that the application has issued more memory requests and several pages have been allocated. Now assume memory is released so that pages numbered zero, four, and two can be reused (in that order). First, the current value of the anchor is written into page-table entry zero, and the anchor points to page-table entry zero. Then page four is released. This is the last page of the region, pointed to by *lastused*, so *lastused* is simply decremented by page size. Page two is released, and this page-table entry is the new head of the linked list of free pages; see Figures 9 and 10.

If the number of used pages in the region drops to zero, all pages have been released. *lastused* points to the beginning of the region, and the anchor is set to zero. In other words, the region is reinitialized.

Full Page Requests

If a memory request is rounded up to a full page, a free page is obtained from the region as previously described. The size field of the page-table entry is set to eight, indicating that the page hasn't been divided into blocks. If the page is later released, its size indicates that it can be returned to the memory region immediately.

Overhead

The space overhead is very low. The page table consists of four bytes per entry. For a 256-MB region, 65,536 page-table entries are needed, assuming the page size is four KB. This equals 256 KB or 0.00098 percent. The size for the freelist and the other internal counters and pointers can be neglected. This amounts to less overhead than you get with heaps maintained by traditional linked lists.

Performance and Memory Utilization

I conducted performance tests for the BSD 4.4 algorithm, the FMA, and the AIX 4.1.4 system heap. The sequence of memory requests and releases and the sizes allocated per request differ from application to application. There is no pattern that is valid for all possible scenarios. The test program (available electronically), therefore, randomly requests and releases previously allocated blocks. The requested size is also determined randomly, within the following limits:

- A block is never larger than the page size.
- Block requests of a particular size may not exceed the boundaries shown in Table 1.

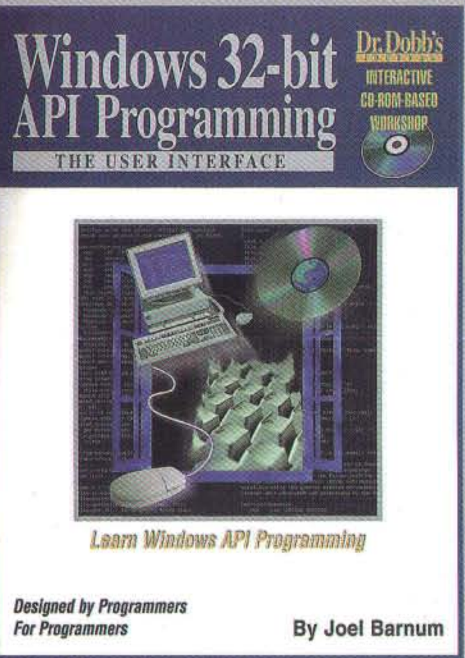
Block size in Bytes	Percentage
16	10
32	30
64	30
128	15
256	6
512	5
1024	2
2048	1
4096	1

Table 1: *Percentage of memory block sizes.*

Learn to be a proficient Windows API programmer without wading through a stack of manuals or spending mega-bucks attending classes or seminars!

WINDOWS 32-BIT API PROGRAMMING

The User Interface



Price: $59.95

Call Now! 800-992-0549

E-mail: orders@mfi.com

Fax: 785-841-2624

Mail:
Dr. Dobb's CD-ROM Library
1601 West 23rd, Ste. 200
Lawrence, KS 66046-2703

International: Use mail, fax, e-mail, or call 785-841-1631

Dr. Dobb's CD-ROM LIBRARY

Dr. Dobb's Journal is excited to introduce a fundamental book for all Win32 API programmers — *Windows 32-bit API Programming - The User Interface.* This book w/CD-ROM will help you learn Windows 32-bit API programming quickly and proficiently, using an intuitive tutorial that takes you through a series of interactive CD-ROM-based workshops. This hands-on workshop enables programmers to visualize concepts, as well as listen to audio explanations for each visual, with pop-up windows containing the voice-over text. Also included are comprehensive hands-on lab exercises and quizzes to test your comprehension.

This compelling tutorial provides the challenge and level of quality that you have come to expect from *Dr. Dobb's Journal.* **The contents of *Windows 32-bit API Programming* will teach you how to:**

- **Create menus and toolbars**
- **Program dialog boxes and dialog controls**
- **Program common dialogs**
- **Program property sheets and wizards**
- **Program listview and treeview controls**
- **Use the Microsoft Developer Studio to create an icon and associate it with a window class**
- **Write a complete winmain function**

System Requirements:

Windows 95 or Windows NT 4.0 (or later)

10MB free disk space

Microsoft Visual C++

Windows-compatible CD-ROM player (double speed or better)

Visit the *Dr. Dobb's* web site for more product information! www.ddj.com/cdrom/

(continued from page 82)

Ten percent of all calls request a block size of 16 bytes, 30 percent request a block size of 32 bytes, and so on. The test program keeps track of up to 5000 allocated blocks. If the table overflows instead of randomly releasing blocks (which can happen due to the way blocks are selected for release), all allocated blocks are freed before the next block is allocated. This reflects the behavior of many applications that allocate memory blocks to build tree or list structures and never release them. They rely on the operating system to release the requested memory when the program terminates. I ran the test program on an IBM Thinkpad 850 with a PowerPC 603e processor running at 166 MHz and 96 MB of memory. The operating system AIX 4.1.4 was in single-user mode.

Table 2 lists the difference in run time when a heap-memory management system is active. First, a sequence of memory requests (including size) and releases is calculated. This sequence is executed twice — first, without actually issuing memory-heap calls; and second, executing memory-heap calls. Table 2 shows the time difference (in milliseconds) between both executions. Table overflow indicates how often the application has freed all allocated blocks before requesting a new one.

As Table 3 shows, the FMA achieves far better memory utilization than the BSD 4.4 allocator. The FMA saves about 49.5 percent of memory when allocating up to 60,000 memory blocks. The percentage of memory savings decreases as the number of blocks allocated increases. This is due to the fact that the test program only keeps track of 5000 allocated blocks.

Interfaces

I haven't altered the traditional C interfaces *malloc* and *free,* or the C++ interfaces *new* and *delete,* for dynamic memory allocation.

The FMA achieves far better memory utilization than the BSD 4.4 allocator

Applications can be developed and tested with standard tools and debuggers, and some vendors offer special heap debuggers. When finished, the operators for *new* and *delete* or the functions *malloc* and *free* are simply replaced.

The C interface consists of the following function calls:

- *int fmainit(size_t, int, char);* creates and initializes the region. The first parameter specifies the size in bytes. The second parameter determines the characteristics of the heap. The region can be allocated in shared memory or in the process's private data segment. Shared memory is mandatory if the heap is to be shared between processes. Information on whether access is private or shared is located in a shared-memory segment. If access is shared, memory requests are serialized (to protect internal data) using semaphores. The region can be pinned in memory. However, certain restrictions apply, such as permissions (root user) and real memory restraints. The third parameter indicates that the region has to be created (if it doesn't already exist).
- *int fmadown();* removes access to the heap. No further requests can be made after this call. Pointers into the memory region may be invalid and could cause core dumps. If the last process sharing the heap calls *fmadown,* the shared segment is removed from the system.
- *void *fmaalloc(unsigned long);* requests a block of memory of a particular size. The address of the block is returned (or null is returned if no more memory is available).
- *void fmafree(void *);* returns a block. The parameter is the address of the block as returned from *fmaalloc.*

C++ interfaces use a structure that must be defined before the first call to *new*; see Listing One (listings begin on page 87).

The first call to *new* creates the memory segment according to the values defined in the structure. The members *size,*

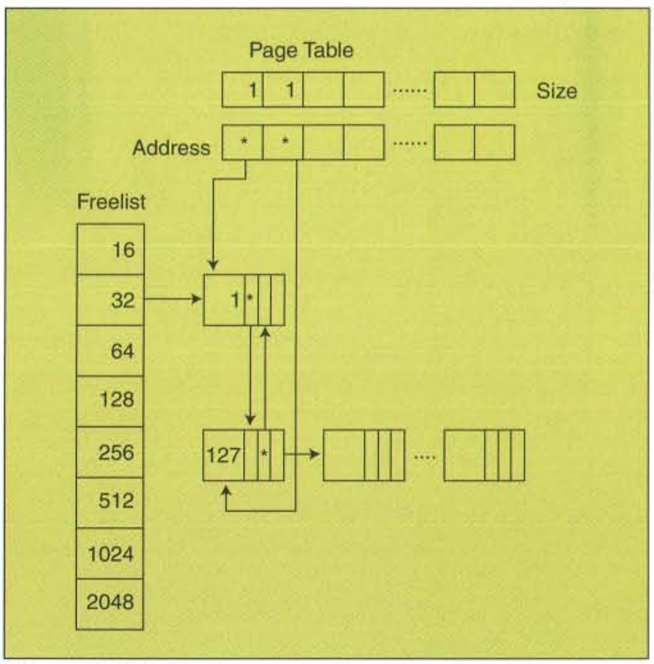


Figure 8: *First block requested is returned.*

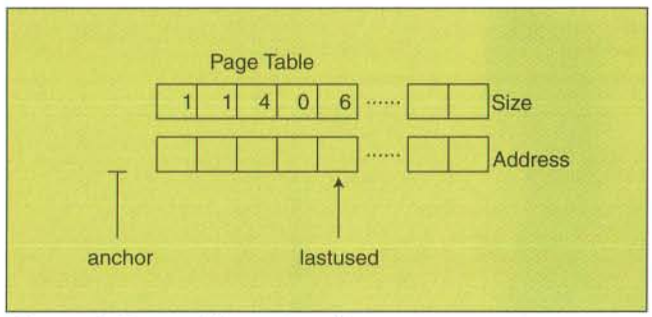


Figure 9: *Page-table entries after more requests.*

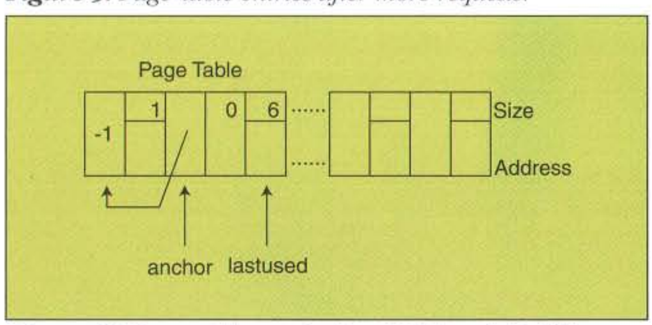
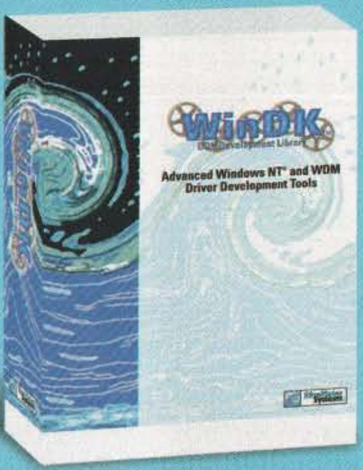


Figure 10: *Page-table entries 2 and 0 form linked list.*

Advanced

Driver

Development



When the pressure is on to deliver mission-critical drivers, you can't rely on ordinary DDK tools alone. You need hard-core tools designed for serious system-level development. You need WinDK™ and NuMega SoftICE™ — the most advanced tools for Windows NT® and WDM Kernel Driver Development.

WinDK 2.5 cuts your driver development time with its proven Driver Development Library. It breaks down the DDK barriers with thousands of lines of encapsulated code and in-depth documentation. WinDK supports C and C++ driver development, as well as enhanced WDM support for USB and FireWire®. With WinDK your driver will be up and running fast.

NuMega SoftICE 3.22 gives you the absolute system-wide control and visibility you need to debug multiple processes, multiple threads, and step seamlessly from user-mode into kernel-mode system calls and back again. DDK tools simply cannot match the reliable debugging power of NuMega SoftICE—The Advanced Windows Debugger™

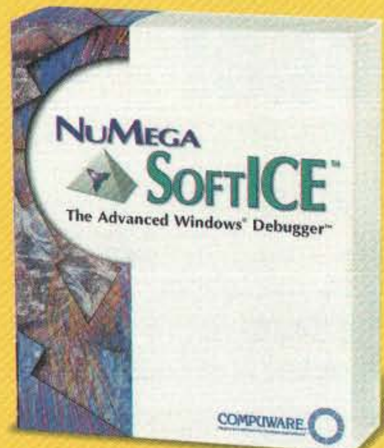
Get WinDK and NuMega SoftICE
The Mission Critical Team.

WinDK/SoftICE Bundle: $999

Regularly $1564, Limited time offer US and Canada only

1-800-962-2114

www.bluewatersystems.com



BlueWater Systems Inc.

Compuware® NuMega™

Advanced

Driver

Debugging

FOR MISSION CRITICAL DRIVERS

30-DAY MONEY BACK GUARANTEE

Copyright © 1998 Compuware Corporation. All rights reserved. WinDK and the WinDK logo are trademarks of BlueWater Systems, Inc. All other trademarks are the property of their respective owners.

(continued from page 84)
flags, and *key* have exactly the same meaning as in the *fmainit* call.

While the first call to *new* creates a memory region (or attaches to an existing one), the last *delete* should delete or detach from it. Since automatic detection of the last *delete* is not possible, a little help from the application is needed. When the member *remove_now* is set to True, the next *delete* call will detach from the region and destroy it if the application was the last one using it; see Listing Two.

If the program is compiled with the "define FMA" set, it will use the FMA, and must be linked with the FMA version of operators *new* and *delete*. Otherwise, it will use the standard C++ operators supplied by the vendor's library.

Blocks Requested	FMA	BSD 4.4 (Run-Time Difference)	AIX Heap	Table Overflow
5000	12ms (2400)	8ms (1600)	82ms (16400)	0
10000	32ms (3200)	20ms (2000)	206ms (20600)	0
15000	62ms (4133)	48ms (3200)	366ms (24400)	1
20000	74ms (3700)	58ms (2900)	502ms (25100)	1
25000	98ms (3920)	72ms (2880)	578ms (23129)	1
30000	126ms (4200)	100ms (3333)	706ms (23533)	1
40000	152ms (3800)	134ms (3350)	938ms (23450)	2
50000	192ms (3840)	160ms (3200)	1198ms (23960)	3
60000	254ms (4233)	190ms (3166)	1446ms (24100)	3
70000	268ms (3829)	200ms (2857)	1604ms (22914)	4
80000	328ms (4100)	240ms (3000)	1810ms (22625)	5
90000	348ms (3866)	278ms (3089)	2102ms (23355)	5
100000	376ms (3760)	320ms (3200)	2240ms (22400)	6
150000	644ms (4293)	472ms (3146)	3356ms (22373)	9
200000	784ms (3920)	642ms (3210)	4512ms (22350)	12
Average per Block	3812 (139.33%)	2736 (100%)	22712 (830.12%)	

Table 2: *Average run time depending on number of blocks requested.*

Conclusion

The FMA is about 40 percent slower (on average) than the BSD 4.4 allocator, but offers the advantage of uniting released memory blocks and reusing pages, resulting in memory savings of nearly 50 percent. The AIX 4.1.4 heap allocator is listed as reference only. It is a general-purpose allocator that sacrifices speed for space optimization. Real-time applications need deterministic behavior, so the upperbound for memory allocation (FMA:4300ns, BSD4.4:3400ns, as shown in Table 3) is of particular interest.

As it turned out, in my example, there was no need to implement support for memory requests spawning several pages.

References

Kelly, David A. *AIX/6000 Internals and Architecture.* New York, NY: McGraw Hill, 1996.

Kernighan, B.W. and D.M. Ritchie. *C Programming Language.* Upper Saddle River, NJ: Prentice Hall, 1978.

Knuth, Donald E. *The Art of Computer Programming, Volume 1.* Reading, MA: Addison-Wesley, 1973.

Master your code! Object Master™ Professional Edition

Develop code the way you think

- No need to know the physical layout of code
- Develop in terms of classes and methods, not files

Quickly analyze and understand existing code

- New team members can quickly learn code architecture
- Master complex framework class hierarchies
- Reverse engineer code
- Easily navigate and manage large projects

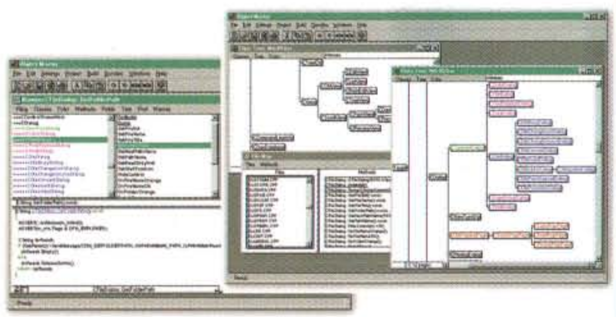
Dramatically Improve Productivity

- Dynamic and customizable code navigation tool and editor
- Browse without having to compile your project

Use the latest compilers and languages

- Integrates with VC++, CodeWarrior, Symantec, Borland...
- Supports C/C++, Java, Pascal, Objective C and more...

AD LINK 371



"The Swiss Army Knife for Code"

C/C++ Java Pascal Objective C

"Object Master has increased my productivity more than any other single tool. There are other browsers out there, but none with the subtle differences that make Object Master such a powerful tool. Object Master is truly in a class by itself"

Doug Ahmann, Adobe Systems

Visit www.altura.com for a FREE demo

ALTURA SOFTWARE, INC.

Altura Software, Inc • 510 Lighthouse Ave. Suite 5 • Pacific Grove, CA 93950
Voice: (408)655-8005 • Fax: (408)655-9663 • email: OMInfo@altura.com

Number of Blocks	Number of Used Pages FMA	BSD 4.4	Savings (in %)
5000	41	79	48.01
10000	82	163	49.69
15000	120	248	51.70
20000	153	295	48.14
25000	190	354	46.33
30000	207	404	48.77
40000	264	499	47.10
50000	314	584	46.24
60000	281	696	59.63
70000	444	790	43.80
80000	490	879	44.26
90000	519	916	43.35
100000	557	993	43.91
150000	758	1255	39.61
200000	935	1461	36.01

Table 3: *Memory utilization in page blocks.*

Leffler, Samuel J., Marshall Kirk McKusick, and John S. Quarterman. *The Design and Implementation of the 4.3 BSD UNIX Operating System.* Reading, MA: Addison-Wesley, 1988.

Vahalia, Uresh. *UNIX Internals: The New Frontier.* Upper Saddle River, NJ: Prentice Hall, 1996.

DDJ

EMBEDDED SYSTEMS

Listing One

```
struct t_heappolicy {
    unsigned long size;
    int flags;
    char key;
    int remove_now;
};
```

Listing Two

```
int main(int argv, char **argv)
{
#ifdef FMA
    t_heappolicy policy = { 1 << 20, 0, 1 };
#endif
    int *ip = new int;  // Create a private heap of 1 MB
    .....
#ifdef FMA
    policy.remove_now =  1;
#endif
    delete ip;
}
```

DDJ

Ultimate TCP/IP
Your Ultimate Answer

- SMTP/POP3
- MIME
- NNTP
- FTP
- HTTP
- DNS
- MFC
- C++
- ActiveX
- Clients
- Servers
- NT Services
- Firewall Friendly
- Custom Protocols
- Multi-threaded Capable
- Royalty Free

Special pricing for Team licenses available.

Developing high quality, Internet enabled applications can be a difficult task. Difficult if you're not using Ultimate TCP/IP that is.

Ultimate TCP/IP was designed to make your job as an application developer easier. Whether you're writing an e-mail enabled application with MIME support, or a custom TCP/IP server to disseminate information throughout your Intranet, Ultimate TCP/IP gives you the tools you need.

Ultimate TCP/IP is very easy to use and integrates painlessly with new and existing development projects.

Ultimate TCP/IP includes 100% full source code, designed so you can easily modify, enhance, and understand it.

Ultimate TCP/IP also includes a convenient collection of drop-in ActiveX components for those development projects that don't require the high power of C++.

The Ultimate TCP/IP Enterprise edition includes an Ultimate pocketTCP™ toolkit for Windows CE development.

Ultimate Guarantee

It's simple. We guarantee your satisfaction. We guarantee that you will have full access to our tech support for a year. If you find Ultimate TCP/IP doesn't fit your needs, we provide a 30-day money back guarantee.

DUNDAS SOFTWARE

1.800.463.1492
sales 416.239.7472 **fax** 416.239.2183
email sales@dundas.com **www** www.dundas.com

AD LINK 279

Active Data Objects & ASP

Executing scripts on a server

Mark Betz

There are times in web development when you need direct access to data from the web server. Perhaps your application is a simple catalog or report generator. But even a complicated extranet app has its two-tier elements. One clear example is the need to manage application state. Not long ago, just about any web app that did more than display pages had to manage its state on the server side. This is still the case if your app needs to be compatible with absolutely every browser ever made. If you're willing to thin the herd and select a subset of clients, then you can move some processing, and thus some state management, out to the client in the form of scripts, controls, and applets. Still, it is difficult to carry state across pages on the client. If, for example, you have an HTML element that may be in one of two states, then its state is probably going to be set on the server every time you build a page. Otherwise, you would somehow have to persist the state on the client across requests. This is certainly possible, but the available solutions may involve things like writing to the local registry or file system, which makes users nervous.

It's easy, however, to carry state across pages on the web server. Early applications for the Web used CGI programs that sometimes wrote to local files or more-commonly accessed data on a server somewhere. The stored data had to be keyed to a particular user (not a trivial problem in HTTP), but otherwise, there was no limit to what could be persisted to the DBMS. CGI programs are probably still the most common means of activating web applications. However, recent technologies focus on extending the web server through the use of componentry in the form of COM, CORBA, and Java objects, and, very recently, on the

use of server-side scripting environments. Server script is particularly suited to cementing the incoming HTTP request to the layer of object interfaces that implements transactional capability. It is also very useful for managing application state and formatting pages. Page generation is handled more readily in script than in CGIs (Perl aficionados will complain, quite rightly, but Javascript and VBScript are certainly more approachable for many developers than Perl). Furthermore, the pages in a web application are subject to frequent change, and scripts provide a very malleable interface between the pages and the less-mutable parts of the system.

Microsoft's Active Server Pages

Microsoft's Active Server Pages (ASP) provides an environment within which a script executes on the web server, resulting in output that is fed back to the client. ASP is compatible with various scripting languages through the use of COM interfaces that expose the scripting engine. Currently, the environment supports VBScript and Javascript. Active Data Objects (ADO) is another Microsoft technology, one built on top of OLEDB, which is in turn built on top of the Open Database Connectivity (ODBC) Standard. ODBC provides a functional interface that separates data access from data providers, and allows applications to execute SQL queries against different server implementations. OLEDB provided the COM interface to ODBC, and ADO simplified OLEDB with an automation wrapper that was useful to Visual Basic programmers. With the growing popularity of ASP as an extension tool for Internet Information Server (IIS), ADO has emerged as a fast, reliable way of connecting your server script to data. Keying application state to a specific user is a good example of using ADO with ASP.

Identifying a user so you can grab the right data is a problem, because HTTP is a connectionless protocol. It is implemented as a set of atomic request/response transactions. The protocol does not provide any concept of a client across requests. In the context of static pages, this is a reasonable choice: It makes the HTTP protocol much simpler, and thus much more robust. However, the lack of connection state has a profound impact on the design of information- or transaction-driven web applications. Consider the problem of authentication across two requests. On request one, the server can receive the user authentication tokens as arguments in a form POST. The tokens can be validated against a user database and, if the user authenticates, the next page in the app can be returned. Request two, originating from the returned page, must

Mark is president of Incurrent Corp., a specialized commerce service provider and developer of Internet cardholder service applications for the payment card industry. He can be contacted at mbetz@incurrent.com or http://www.incurrent.com/.

somehow convey to the server the authenticated state, as well as the identity, of the user. When the server responds to request two, it will use this information to determine the current state of the application.

There are basically two ways to get user information across the connection:

- Piggyback it onto the HTTP transaction, either as parameters to a GET, or name-value pairs in a POST.
- Use cookies. Cookies are short strings of text that are generated on the server and sent to the client, and are then returned to the server by the client when a request is made.

ASP provides an automation object named *Session* that utilizes cookies to maintain connection state. The *Session* object is very easy to use. However, it suffers from problems with cookies, namely that users perceive them as a privacy threat and can disable them. Cookies were the only means Microsoft had for enabling user sessions without intruding into the application script. The first method, adding user information to GETs and POSTs, is more intrusive, but more reliable. Since the script-level requirements can be minimized through the use of server includes—blocks of script included into a page prior to script execution—or by encapsulating the needed functionality in COM controls, developers often prefer this method. It also makes a more interesting example of ADO use.

In this example, there is a set of pages that cannot be retrieved by users unless they have authenticated. Once authenticated, a user will be assigned a session key, and the key will be used to access application state on the web server. The session key will be embedded into returned pages as a parameter to any URLs, and as a hidden input field in any forms. On the server side, the script has the responsibility of making sure that a controlled page is accessed with a valid session key. If a request is received with no session key, or an invalid session key, then the user must be redirected to the login page. What constitutes an invalid session key? The system may have been forced to invalidate the key due to an error. A more common reason is simply that the user is finished, and no longer needs the key. Unfortunately, there is no easy way to know when this is the case. HTTP provides no hint of state across requests, so how do we know which request was the last one? A logoff button can be provided, but users can easily surf out without clicking it. The only reliable mechanism involves a process that times out inactive sessions. We won't get into that aspect of the system very deeply here.

Our database requirements are simple. Figure 1 shows two tables. The first is named *tbl_User*. This table has four columns. *UserID* is an IDENTITY column that serves as the primary key. *UserName*, *UserPwd*, and *UserFullName* are all *char* columns with self-evident uses. The second table is *tbl_Session*. It has three columns. *SessionID* is another IDENTITY column providing the primary key. It isn't necessarily a good idea to use an IDENTITY column to generate session keys. A better mechanism is to generate a pseudo-random 32-bit integer, then hash it against the client's IP address. For this example, we'll keep it simple. The other two columns of *tbl_Session* are *UserID*, a foreign key into *tbl_User*, and *LastAccessTime*, which is a *smalldatetime* used to control inactive sessions. These two tables will provide all the storage we need for the purposes of authenticating and providing session state. In practice you'd have other tables keying on the session or user IDs, and containing various parts of the application's state.

Listing One (listings begin on page 111) provides examples of the first two components in our app—the login form and the login script. When the user clicks the login button in userlogin.htm, a POST is executed against the web server, containing the form input field contents, and identifying the login.asp script as the target. On entry to the login.asp script, we first need to extract the login parameters from the HTTP request. ASP provides an automation object called *Request*. Among the *Request* object's several collections is one called *Form*. The values in the *Form* collection correspond to the named input fields in the login form. Thus, the script *<%UserName = Request.Form("UserName")%>* retrieves the value that the user typed into the "UserName" edit. The first duty of login.asp is to extract the user name and password from the *Form* collection, and get them into variables. If this operation fails for any reason, the user must be redirected back to userlogin.htm. Checking parameters before processing them is always a good idea: The "calls" into your script are often URLs that hang out in the browser address bar, tempting users to play around with them. It's a good idea to program defensively.

Once the parameters are verified, we need to create the database connection. ADO's *Connection* object represents a connection to an ODBC datasource. Opening the database connection requires three steps.

1. First, the connection string is created. It contains the ODBC datasource name, the name of an authorized user, and the user's password.
2. Next, the connection object is created. There are two primary means of creating objects in ASP. Either the object is inserted into the page using the HTML OBJECT tag and the RUNAT="SERVER" parameter, or the *Server* object is used as a factory. The *Server* object is one of the built-in ASP objects, and its *CreateObject* method can create an instance of any scriptable automation object. It's a bit cleaner than the OBJECT tag, so it is the path we'll take. The only parameter we need is a string identifying the class and interface we want—in this case, the ADODB class and the *Connection* interface. The result of the call is assigned to the *Conn* variable.
3. Finally, the *Connection* object's *Open* method is called and passed the connection string we built previously. If all is successful, when this method returns, the *Conn* variable holds an open

table tbl_User				
	int UserID	char(25) UserName	char(25) UserPwd	char(50) UserFullName
Identity	✔			
Primary Key	✔			
Foreign Key				
Nulls				

table tbl_Session			
	int Session ID	int UserID	smalldatetime LastAccessTime
Identity	✔		
Primary Key	✔		
Foreign Key		✔	
Nulls			

Figure 1: *Sample database tables.*

connection on our datasource. Should an error occur anywhere in this process, the ASP engine will cease evaluating the script and return an error to the client.

With the connection open, the first order of business is to authenticate the user by looking up the user name and password in the database. A simple SQL SELECT statement suffices in this example. If no record containing both the name and password is found, then the authentication fails and the user would normally bounce out to an error page. In this case, we'll just return them to the login screen. To execute the SQL statement, we first build it into a string. This process is familiar to anyone who has programmed SQL Server using Microsoft's APIs. Once the query string is constructed, it is passed to the *Execute* method of the *Connection* object. The *Execute* method is capable of running any SQL statement that is supported by the data provider. Since ours is SQL Server, we've got a wide range of functionality available. Later, we'll see an example of compound statements, multiple result sets, inserts, and so forth. If you're keeping an eye on portability, you don't want to get too fancy, and in any case, it is evident from reading the listings why restricting the use of ADO and ASP to managing application state and other local web server tasks is a good idea. ADO enshrines the data schema in the application source code.

The result of a successful call to *Execute* is an interface on a *RecordSet*. The *RecordSet* object is ADO's primary interface to data on a server, with a number of properties and methods that provide full cursor access to result sets. Note that not all operations are available from every data provider. Often, if an operation is not supported, the result is a NULL, or other ambiguous value. In the case of the *AbsolutePosition* property, for example, a return value equal to the constant *adPosUnknown* may mean the *RecordSet* is empty, the current position really is unknown, or the provider does not support the absolute position property. The moral is: Know your datasource. If you're keeping things relatively simple, you won't have any problems. At the moment, our needs are very simple. If *Conn.Execute* completes successfully, then the *RecordSet* is assigned to the variable RS, and the script in login.asp checks to see if the RS.EOF property is True. EOF is set to True when the current position is one after the last record returned, and so is True for an empty result set. True, in this case, means the user record wasn't found, so we send the user out to userlogin.htm again. If RS.EOF is False, then we retrieve the selected *UserID* into a variable by querying the *RecordSet Field* collection. Fields are available for the current row and may be accessed by name or ordinal position. In this case, we ask for *UserID* by name.

With the user authenticated and the database ID in hand, the script has one more check to make before creating the session. Since this is a web app, the user may be trying to log on again while a session is still active for him. To prevent this, the script queries the session table to see if the *UserID* is present. If it is, then the selected session key can be reused in the redirect to the welcome page. Assuming the user does not have an active session, the script must create one by inserting a new record into the session table. We noted earlier that SQL Server was going to generate the session keys for us through the use of an IDENTITY column in *tbl_Session*. An IDENTITY column is automatically stuffed with a unique value when an INSERT takes place on the table. Since we need the session key, we also need to get our hands on the value of that column right after the INSERT. Fortunately, SQL Server supplies the @@IDENTITY global that contains the last generated value for an IDENTITY column. Unfortunately, @@IDENTITY is only valid until the next INSERT against a column without an IDENTITY attribute occurs. We don't know when that will happen. Therefore, we'll use a compound statement to insert the session record and grab the IDENTITY column value all at once.

The use of a compound statement gives us a chance to look at how the *RecordSet* object handles multiple result sets. The statement itself is simply a batch of commands separated by semicolons. The first command in our batch is an INSERT. What we're inserting is the *UserID* and the initial value for *LastAccessTime*. SQL Server generates the SessionID, updating @@IDENTITY to the new value, which is selected out by the second command of the batch. Assuming *Conn.Execute* completes successfully, RS will contain an empty *RecordSet* resulting from the successful INSERT. Since we executed a batch, we know that there is another *RecordSet*. To get to it, we call *NextRecordSet* on our existing *RecordSet*. What we get back, hopefully, is a new *RecordSet* that is positioned at the first row in the results from the next command in the batch. At this point, all that remains is to extract the *SessionID* by querying for RS(0). We use an ordinal here because our SELECT did not assign a column name to the result. RS(0) accesses the first field of the current row, which is our selected identity value. The last step in the process simply redirects the browser to the welcome page with a URL that includes the session key.

The welcome page makes its way into our example because it demonstrates the basic tasks that all session-controlled pages must accomplish when a user requests them: checking for a valid session, updating the session's *LastAccessTime* column, and using the validated session to retrieve user information. All of these things happen in welcome.asp, which is presented in Listing Two. In essence, this file is a template from which any other session-controlled page can easily be built. Normally, we would take the session validation script and move it out to an include file that can be parsed into any page that needs the ability to verify a session. Once again, though, we'll try to keep it simple for the purpose of illustrating how ASP and ADO work together to handle these tasks. The welcome.asp page also demonstrates the combination of HTML and ASP script that is the strength, and the weakness, of server-side scripting. The strengths lie in the ease with which output can be generated and fed back to the client, using all of the power of Visual Basic's string handling. The weakness is that it can be difficult to combine these elements in a way that leaves a structured and readable source file. Fortunately, in this case, nearly all the script precedes the HTML in the file.

In this script, as with the login script, the first thing to do is certify that the required parameters are present in the query string. In a more complicated application, we would probably have to do some validation on the types and ranges of the parameters as well. Fortunately, all we are concerned with here is that the *sesskey* argument is present. It doesn't really matter what the type of its value is. One of the nice things about working in a script environment where all variables are of type VARIANT is that you can almost always compare what you have against what you want. If we want a numeric session key and the user edits the URL to present a string instead, the result is the same as for any other invalid session key. (Of course, working with nothing but VARIANTs causes lots of problems, too, but that's a topic for another time.) Once the script has determined that the session key is present, it has to validate the session. A session key is valid if it is the primary key to a record in the session table, and the *UserID* column of that record is a foreign key to a valid record in the user table.

We could easily wrap these two requirements into a single query, by joining the *tbl_User* and *tbl_Session* tables on the *UserID* column. We won't do this because it requires that the data provider support joins. The business logic of a real application may need to set and enforce this requirement, but we prefer to keep the

simpler application-state management as portable as possible. Before executing the statements to validate the session, the script creates a database connection exactly as in the login script. This is the kind of logic that can be extracted into an include file, since the same connection string is often used throughout an app. It might not be a good idea if, for example, the user also has database privileges, and you want to build the actual user ID and password into the string. In that case, you'd probably build the string when the session was created and store it. Our example envisions a case where the application and database security models are separate. Regardless of where the user's ID and password come from, there are other parameters that you shouldn't have to retype into every page. In practice, we've found it handy to store information such as the datasource name, login ID, and login password in the system registry on the web server or another host on the back-end. A simple COM control can extract this information into script variables, so that the pages can be built generically.

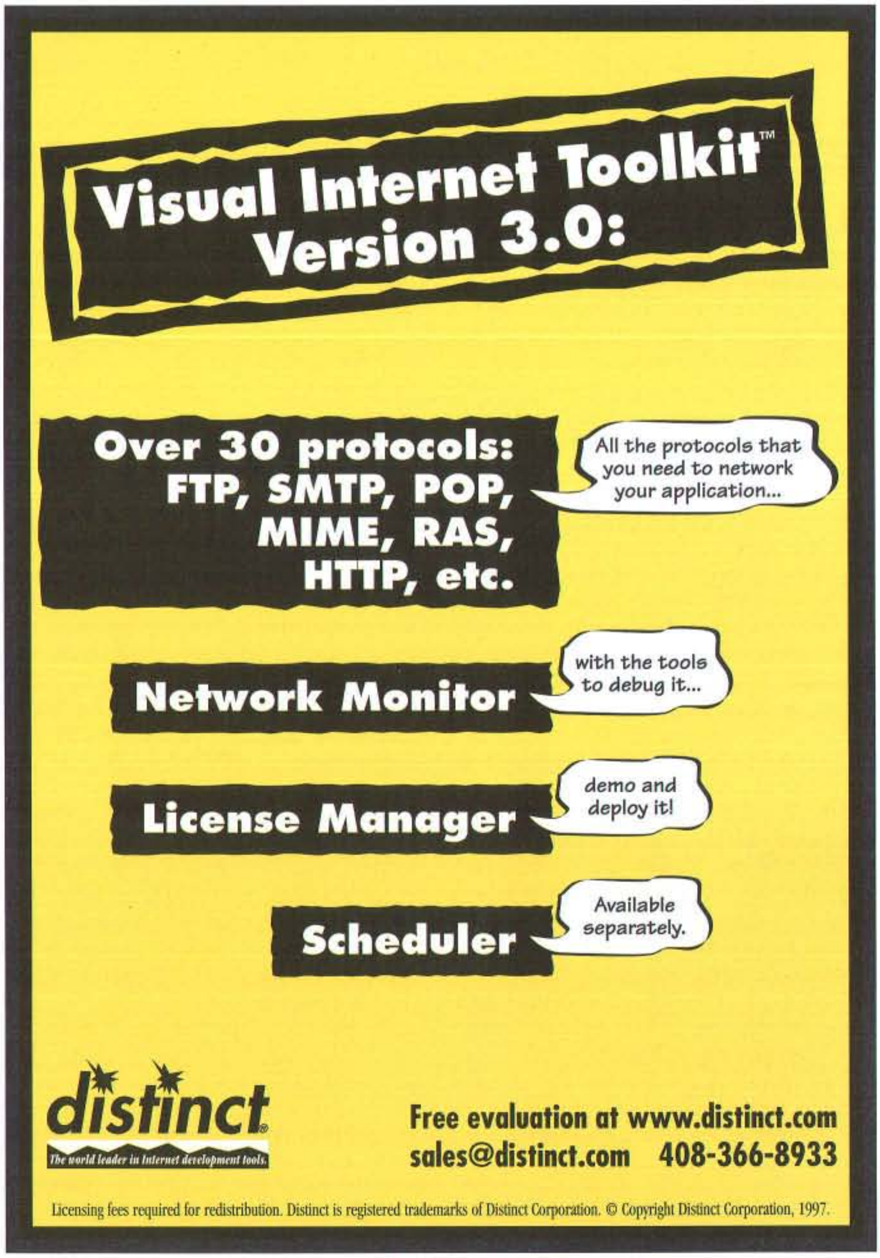
With the connection in hand, the script executes a SELECT to get the *UserID* from the record in *tbl_Session* that is identified by the session key passed in the URL. If this record is not found, then the session key is invalid. If the record is found, the returned *UserID* is extracted from the *RecordSet* object in RS. The next test is for a valid user record. The script performs this test by selecting out the *UserFullName* column, which will be used later in the welcome message. A failure here means an error in the integrity of the database: There should be no records in the session table with a *UserID* column that is not a valid foreign key into the user table. In fact, if we're using a full-featured database such as SQL Server to store the application state, the *UserID* column of *tbl_Session* will be constrained so that only valid foreign keys can be inserted into it. Such a constraint would allow us to select the *UserFullName* column without any fear that the operation could fail due to an invalid user ID. Since we are not assuming such a constraint here, we check to see if the returned *RecordSet* is empty. If it is, then the script makes use of the *Response* object to write an error message to the client. If all goes well, the script completes processing and the variable *UserFullName* is used in the body of the HTML to extract the user's name into the welcome message. This illustrates one of the more powerful ways in which ASP script integrates with the HTML output stream.

Conclusion

Active Server Pages are useful for generating output and managing application state on behalf of a client. When combined with Active Data Objects, your script can manipulate ODBC data sources to do nearly anything that is possible in native SQL. In our applications, we have used ASP and ADO to execute stored procedures, create temporary tables, and wrap up transactions. Still, it is important to recognize this model's limitations, which flow more from the limitations of scripting languages than from limits on what ADO can do. Scripting languages, whether running on the server or client, suffer from the problems normally associated with interpreted, weakly typed languages. Misspell a variable name in VBScript and you've created a new variable. If you're lucky, it will generate an error. ASP suffers particularly from the difficulty of mixing script and HTML in ways that retain structure and maintainability. It is also obvious that too liberal use of ADO in script will lead to a very tight coupling between the application logic and the data schema. This is acceptable for small applications, but not for critical systems. For this reason, we have argued here in favor of using ASP with ADO to manage application state and glue components together. In this role, they serve as one more layer in an architecture that may include dynamic web pages on the front end, and mission-critical object interfaces on the back end.

DDJ

(Listings begin on page 111.)



AD LINK 125

The Hot Views Graphics Library

A graphical user-interface library designed for scientific modeling and simulation

David P. Heddle

The Hot Views (Hv) graphical user-interface library was designed for use in scientific modeling and simulation applications. It is layered on top of the ubiquitous UNIX-based X, Xt, and Motif libraries. Hv will likely be useful to you if:

- You are developing a model or simulation of real objects that you would like to display using a (floating-point) world-coordinate system, and/or you need to generate scientific/engineering style graphs of simulation results.
- Your application requires pointer feedback; when a user points at a representation of an object on the screen, you want to provide instant visual feedback regarding the object's state.
- You want to be able to print the graphical representation of the simulation, or render it to a Postscript file for import into another document.

These may sound like trivial requirements until you consider that X, Xt, and Motif do not provide such services. They are pixel based, making representations of real objects tedious. Nor do they provide direct information regarding which object contains the pointer. Finally, they do not support the translation of low-level X-based drawing into Postscript.

The world-coordinate systems maintained by Hv are a more substantive feature than you might first suspect. You can easily render floating-point coordinates onto a pixel-based display by simple scaling. However, Hv will automatically handle the arbitrary linear transformations associated with zooming, scrolling, and object rotation. Furthermore, Hv does not indiscriminately transform every object. For example, suppose you placed a "print" button on top of a map. If you zoomed into the map, Hv would transform the borders accordingly but the button would stay the same size. Other Hv features include drag-and-drop, integrated plotting, zooming, time-step simulations, animation, drawing tools, double buffering, virtual desktop, online and balloon help, map drawing, drawing primitives, scrolling, simplified fonts/colors, 3D sculptured look, image menus, and timed redraws.

Hv's drawing is optimized to redraw the minimum amount of a window that needs refreshing. Hv also lets you earmark items for off-screen caching. For example, a war game might cache maps, which are expensive to draw. With Hv, as tokens are dragged over the map, the user doesn't have to wait for the background maps to be recomputed and redrawn.

Hv is comprised of about 60,000 lines of ANSI C code. Applications have been written in C/C++. It has been tested on virtually all flavors of UNIX, including Linux. The Linux version has been tested with commercial Motif libraries as well as the recently available and cleverly named "lesstif freeware Motif clone."

Hv was developed at the Thomas Jefferson National Accelerator Facility, a Department of Energy nuclear physics research laboratory. It is in use at many government and commercial sites worldwide and available via anonymous ftp from ftp.cebaf.gov in /pub/heddle/Hv or at http://www.cebaf.gov/~heddle/Hv. The distribution is replete with demos and a comprehensive programming manual.

The Hv Paradigm

An Hv application is contained in a single X window, called the "main window," with a single menubar, not unlike a Macintosh desktop. Within the main window are additional window-like objects called "views." Views might contain independent simulation displays or may represent different perspectives of a single model. Figure 1 is an example of an Hv main window containing multiple views. This structure was chosen over independent X windows for each view to avoid having to hunt around a cluttered screen for a lost window.

The views behave as you would expect. They can be moved, resized, closed, scrolled, and exploded, within the confines of the main window. However, unlike most windows, the views can be arbitrarily zoomed by rubber-banding a rectangle while the middle mouse button is pressed.

Contained in each Hv view are Hv items. These include Hv versions of common control widgets such as sliders, buttons, rainbow scales, and wheels. There is also a suite of world-coordinate-based items, such as points, lines, rectangles, ovals, text, and polygons. More importantly, views contain user-defined items that represent real-world objects or graphs. Typically, developers only have to provide the code that draws the item (using the Hv drawing primitives). Hv will take care of the item maintenance and Postscript rendering.

Hv provides many items that already exist as Motif widgets, including buttons and other control widgets. This was done partially to provide a common interface; you do not use a Motif API for some objects and an Hv interface for others. This

David is a physicist at the Thomas Jefferson National Accelerator Facility in Newport News, Virginia. He can be contacted at heddle@jlab.org or heddle@jdhtech.com.

Tight Deadlines for Multithreaded Apps?

Beat the clock with Threads.h++!

Your users want responsive apps. **Your boss** wants the project finished on time. **You need Threads.h++!**

Threads.h++ is **your complete solution** for quickly writing portable, multithreaded applications. By encapsulating key multithreading features as C++ objects, Threads.h++ handles the low-level details of multithreading for you, providing even cross-platform compatibility! Just write your applications once, then recompile them to run on Windows, Unix, or OS/2.

Threads.h++ simplifies multithreaded programming by providing high-level threading abstractions such as producer-consumer queues and IOUs, as well as a broad set of classes for thread creation, control, and synchronization.

Want to leverage your single-threaded code? Use the classes in Threads.h++ to multithread your existing code without making the intrusive changes required by other libraries.

Come to our Web site to learn about how Threads.h++ can dramatically speed your apps!

Count on These Key Features in Threads.h++

- ✔ Platform-independent threading API for portability
- ✔ Classes for thread creation and control
- ✔ Complete set of synchronization mechanisms
- ✔ Higher-level threading abstractions
- ✔ Classes for multithreading your existing code
- ✔ Built on a solid foundation: Tools.h++

Enter to win a Palm Pilot™ at our Web site!

www.roguewave.com/ad/pilot

The Software Parts Company™ Rogue Wave SOFTWARE

Call us toll free in the U.S. at (800) 487-3217.

Telephone us in Europe: Rogue Wave Software GmbH: +49-6103-59 34-0 ◆ Rogue Wave Software B.V.: +31-20-416 06 57

Rogue Wave Software S.A.R.L.: +33-1-5568 1008 ◆ Rogue Wave Software-UK. Ltd.: +44-118-988-0224

Random drawing for PalmPilot will be held July 15, 1998. Winner will be notified by telephone. Rogue Wave software employees and their families are not eligible. Void where prohibited by law.

Rogue Wave and .h++ are registered trademarks, and Threads.h++ is a trademark of Rogue Wave Software, Inc. Palm Pilot is a trademark of 3Com Corporation. All other trademarks are the property of Rogue Wave Software, Inc. or their respective holders.

AD LINK 215

(continued from page 92)
also supports widget sets other than Motif. The widespread acceptance of Motif and the fact it is now bundled on all UNIX workstations has allayed this concern. Still, the minimization of the use of Motif may prove beneficial in porting Hv to non-UNIX platforms such as Windows 95.

To summarize, an Hv application has a single main window that confines an arbitrary number of Hv views. Each Hv view is comprised of multiple Hv items. The items come in two basic flavors—control widgets and user-defined representations of real objects.

Hv Views

An Hv view is comprised of three distinct areas:

- The HotRect is the canvas where the representation of the simulation will be displayed.
- The Control area is where application state is maintained and control items are typically displayed.
- The Feedback region is where live updates associated with tracking the pointer are reported.

Figure 2 is a typical view arrangement which designates the three areas. Figure 2 also includes a familiar set of controls for dragging, resizing, exploding, hiding, and scrolling an Hv view.

Hv Programming

A preliminary design should be carried out prior to the development of an Hv application. This involves deciding how many different types of views are required and then deciding (for each view type) what control items are needed, what HotRect (usually user-defined application-specific) items are needed, and what manner of pointer tracking (feedback) would be informative.

The program hvmap in Figure 3 (included as a demo in the Hv distribution) displays political maps on either a Mercator or a globe-like orthographic projection. As part of a preliminary design phase, I decided to use just one view type, providing controls so the user can toggle between the two projections. Additional controls allow users to toggle the display of a latitude-longitude grid and a CPU-expensive rendering of elevation data.

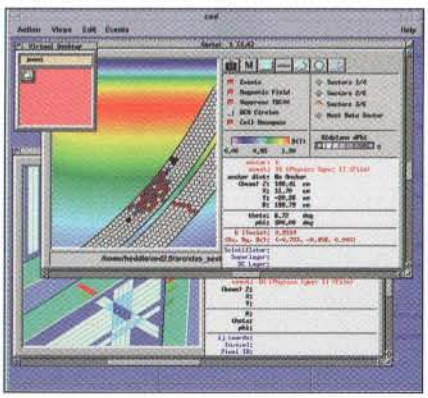


Figure 1: *Multiple Hv views displayed on the Hv main window.*

The alternative was to define two distinct view types, one for each projection. I rejected this as unnecessarily complicated. One view type does not mean there can be only one view. Any Hv view can be indefinitely cloned: I can have any number of hvmap views, each a carbon copy in regard to the controls and feedback, but independently positioned, sized, and zoomed, some showing Mercator projections and others orthographic. This is a common tradeoff in an Hv application—to choose between defining a new view type or adding controls (usually radio buttons) on a base that selects different renderings.

The main program in an Hv application generally consists of three lines: one to initialize Hv, one to call the application's private initialization, and a final call to enter an event dispatching loop. hvmap's main program is presented in Listing One (listings begin on page 112).

The first call handles the Hv and X/Motif initialization. *Hv_VaInitialize*, like many Hv routines for creating views, items, and other objects, uses NULL-terminated, variable-length argument lists, the bulk of which are keyword-value pairs initializing an attribute of the object (or, in this case, process). Variable-length routines all begin with the *Hv_Va* prefix. Hv initialization performs (transparently) all the underlying X and Motif initialization, creates the main window, allocates fonts and colors, and creates the menubar. *Hv_Go* implements an event loop where X events such as pointer motion, pointer clicks, and keystrokes are handled. You don't need to know any of the details of the Hv initialization, or how Hv processes X events.

Variable-Length Argument Lists

The use of variable-length argument lists was indispensable in making Hv programming tractable. The alternative was complicated creation signatures with many parameters or a slew of attribute-setting routines called after an item or view was instantiated. Since this technique may be of general use for any program, I will take a momentary detour to explain how variable-length argument lists were used in Hv.

A variable-length argument list in Hv looks like *Hv_VaX(r1,..., rn, v1,....vn, NULL)*, where *r1* through *rn* are required, *v1* through *vn* are optional, and NULL terminates the argument list. At least one required argument must be provided; you cannot write an ANSI C procedure that has only optional arguments. (However, ANSI C doesn't require NULL termination; you could have one of the required arguments describe the optional arguments that follow. This is how *printf* and related procedures work in C.)

For example, the procedure for creating Hv items has one required argument, the view (of type *Hv_View*) where the item will live. The prototype for the function is *Hv_Item Hv_VaCreateItem(Hv_View View, ...)*. The ellipsis is not representative—you must actually type three periods. You will probably have to include the stdarg.h header file (or the older, non-ANSI sys/varargs.h header file) to use variable-length argument lists. The top of this procedure declares *va_list ap*, then calls *va_start(ap, View)* to initialize *ap* for handling the variable arguments. *View* is the last required variable here. The *va_list* and *va_start* symbols are part of Standard C. Once *ap* is initialized, you can use it as any other variable. In particular, *Hv_VaCreateItem* passes it to *Hv_GetAttributes*, which takes the *va_list* variable and processes it, stuffing the results into an array, which the remaining code in *Hv_VaCreateItem* uses to initialize the item as specified.

The only remaining issues involve how *Hv_GetAttributes* processes the *va_list*. Basically, it pulls off one argument and checks if it is the NULL terminator. If it is not, it assumes it is a keyword and pulls off the next argument as the corresponding value. It places this in a union and interprets the value based on the data type associated with the keyword. Listing Two is the code for *Hv_GetAttributes*.

Designing an Hv Application

When developing an Hv application, the bulk of your work goes into the private, application-specific initialization routine *Init()*. Here is where an application's initial views are created along with the controls, other standard and user-defined items, and entries into the feedback area. Any routine initialization (such as initializing global variables) and menubar additions or modifications are also done here. Listing Three presents the application-specific *Init()* procedure for hvmap.

For example, the *WindowTile* and *AddLogo* procedures use Hv primitives for tiling the main window with a propaganda graphic and drawing a corporate logo on a special welcome view present in many Hv applications. *InitControls* merely initializes some global variables. The *InitQuickZoom* procedure sets up predefined zooms for hvmap, available via a

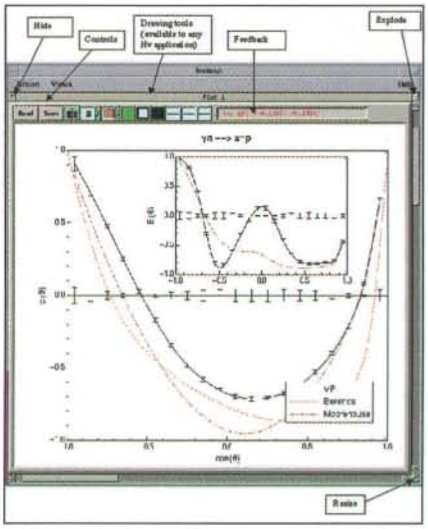


Figure 2: Scientific plot view, which is available in any Hv application.

mouse-button-controlled pop-up menu. These details are best left for you to uncover when building and examining the demos at length. All of these procedures are optional.

Listing Four is the *MakeMenus* procedure, which modifies the default menus provided to all Hv applications. This creates, via variable-length argument calls to *Hv_VaCreateMenuItem*, three menu items and attaches a callback (to be invoked when the user makes that selection) to each. These particular items are attached to predefined menus—Help and Action—present in all Hv applications. You can also create an entirely new menu and attach items to it.

Listing Five shows the *FinalInit* function. *Hv_AddPath* adds a search path that Hv will automatically scan for data files. *Hv_InitMaps* initializes the maps. (The map initialization is an Hv call, *Hv_InitMaps*, because support for drawing maps is built into Hv, not in the hvmap demo.) The interesting line is the *NewView* line, which creates an initial map view. This procedure is too long to present here, but the most important part of it is the embedded call to *Hv_VaCreateView,* which creates the actual view; see Listing Six.

For example, the *(Hv_INITIALIZE, setup)* pair in Listing Six tells Hv to call a private initialization routine after the view is instantiated. In that routine (*ViewSetup*, in this case), the applications will attach the items (as well as the feedback instructions) to the view. Listing Seven creates the radio buttons in Figure 3 that let users toggle between projections. It actually creates a complete set of radio buttons. If a new projection is supported, you only need to add another pair (*Hv_OPTION, "NewProjection"*) to the argument list.

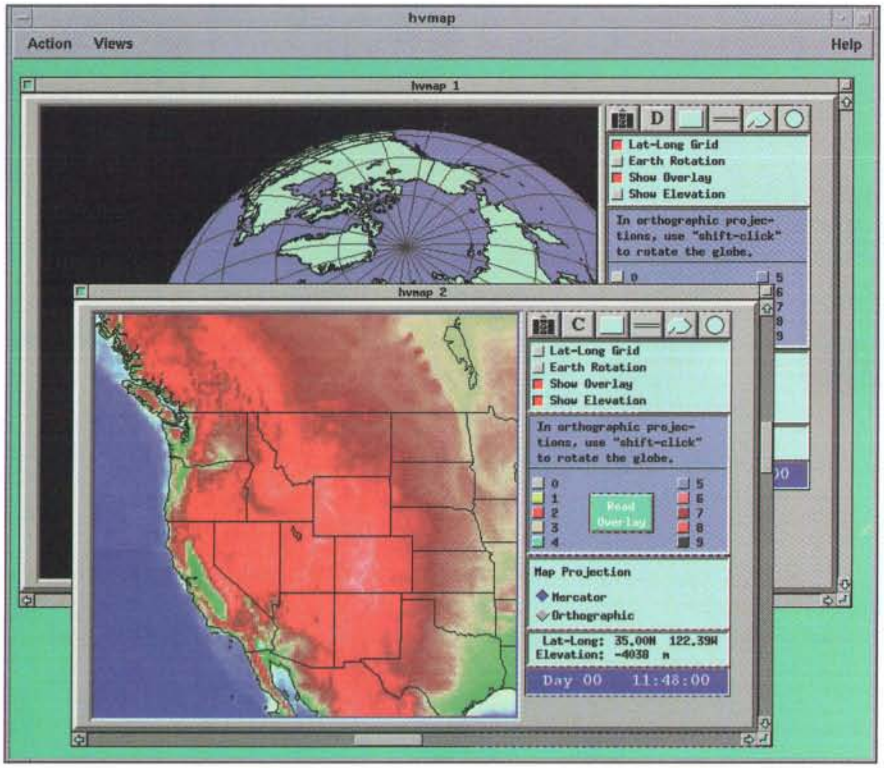


Figure 3: Two views from the program hvmap (one of the demos in the Hv distribution).

Conclusion

When you create views and items, you attach callback procedures that are invoked in response to various actions. For example, you can provide a feedback procedure for a view that Hv will call when the pointer moves within the view's HotRect. You have to convert the pointer position provided by Hv and create a meaningful string for Hv to display in the appropriate position in the feedback area.

There are two ways to create user-defined items. One way is to create truly user-defined items that Hv knows nothing about. This approach requires you to provide a procedure for drawing the item as well as support procedures that tell Hv how the object should respond to scrolling, resizing, rotation, and other events. It's usually better to piggyback on an existing Hv item. For example, many items can often be represented by world-based polygons—those with vertexes stored as world, rather than pixel, locations. Hv has a predefined world-based polygon, so creating an item based on it is advantageous because Hv already knows how to handle all the basic drawing and item maintenance. You need only provide a customized drawing procedure that Hv calls after (or in lieu of) its basic polygon drawing, effectively converting the item from generic to specific.

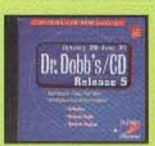
Since hvmap only displays maps, and maps are built into Hv, it has no need to create new items. A more serious Hv application is ced, which is used to display data in a massive (six-meter diameter) nuclear physics detector. One component of the detector is a device called a "drift chamber." Two drift chamber items are visible in Figure 1. They are in the topmost view, sitting on a multicolored background and are tiled with hexagons, some of which are filled. The code that creates one of these drift chamber items is provided in Listing Eight.

This is an example of the piggyback technique. The item is created not as a user item (with *Hv_TYPE* of *Hv_USERITEM*), but as a world polygon, from the predefined suite of Hv items. After instantiation, the type is changed to a user item, and the drawing is redirected to a user-provided drawing routine.

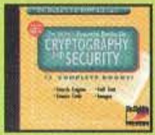
An alternative is to use the *Hv_USERDRAW* attribute to specify a drawing routine. With this approach, Hv will first call its built-in world polygon drawer, then call the user provided routine. By overriding the *standarddraw* pointer (as in Listing Eight), the default world polygon drawer is never called.

DDJ

(Listings begin on page 112.)



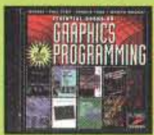
EXPERT ADVICE WHEN YOU NEED IT MOST



Dr. Dobb's CD-ROM Library

ORDER FROM OUR WEB SITE! www.ddj.com/cdrom/



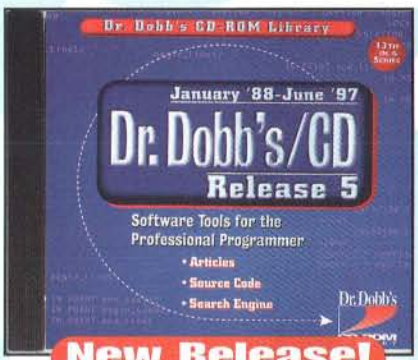





The Software Professional's Essential Resource

The *Dr. Dobb's* CD-ROM Library is an extensive collection of powerful tools and resources, essential to every programmer's productivity and success. The library focuses on critical aspects of programming ranging from an expansive back-issue archive of *Dr. Dobb's Journal* to essential book selections on *Algorithms and Data Structures* and *Graphics Programming*. And coming from *Dr. Dobb's*, you know that each CD-ROM is chock-full of source code. Take a moment to browse through our library. We guarantee your satisfaction.

Dr. Dobb's/CD Release 5



New Release!
Fast Search Engine

Dr. Dobb's/CD Release 5 provides you with immediate and complete access to nine-and-a-half years of *Dr. Dobb's Journal*. Every Algorithm Alley, every Swaine's Flames, every article, and every bit of source code from January 1988 through June 1997 is included on this CD-ROM. In addition, *Dr. Dobb's/CD Release 5* contains all issues of *Dr. Dobb's Sourcebook*.

Bringing together a decade's worth of perspectives and expertise from the most respected programmers in the industry, *Dr. Dobb's/CD Release 5* offers both long-time and first-time readers an unparalleled programming resource.

Use *Release 5*'s powerful search engine to find what you are looking for and then print it, copy/paste it, or directly export it as a file. No more sifting through dozens of back issues, and manually keying source code into your current programming project. With *Dr. Dobb's/CD Release 5*, you have it all when, where, and how you want it!

***Release 5* includes these must-have features:**

Super-fast full text, Boolean, search engine, indexes for both articles and authors, copy/paste/print functionality, save searches for quick access later, annotate and bookmark for future reference needs.

Save $30.00! NEW REDUCED PRICE! $49.95

$29.95 upgrade available to prior owners! To upgrade call: 800-822-1162

ORDER TODAY! 800-992-0549

U.S. & Canada

All Other Countries:
785-841-1631
E-mail: orders@mfi.com
Fax Orders: 785-841-2624

Mail Orders
Dr. Dobb's CD-ROM Library
1601 West 23rd St., Suite 200
Lawrence, KS 66046-2703 USA

Dr. Dobb's CD-ROM LIBRARY

Go to our web site for more products from Dr. Dobb's CD-ROM Library:
www.ddj.com/cdrom/

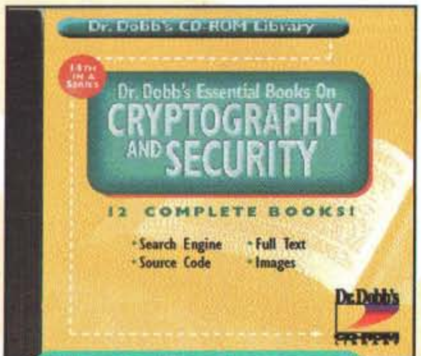
BUY 2 CDS AND GET 25% OFF THE 3RD!

DR. DOBB'S ESSENTIAL BOOKS ON CRYPTOGRAPHY AND SECURITY CD-ROM

The editors of *Dr. Dobb's Journal* have selected the most important books on encryption technology and have compiled it into a resource every programmer must have. This CD-ROM presents both the theory and practice of network security implemented in C, Basic, and other familiar programming languages.

Special Features include: Full-Text Search Engine, Complete text of all books, Hyperlinks across all books, Indexes for all books, Technical bulletins, Security briefs, and Cryptographic FAQs from RSA Data Security.

The *Essential Books on Cryptography and Security CD-ROM* provides detailed documentation and analysis of: Classical and Modern Encryption Algorithms, Private-Key and Public-Key Cryptography, The Data Encryption Standard, Signature Schemes, Block Ciphers and Stream Ciphers, Plus Much, Much, More!



New Release!
Fast Search Engine

Price! $99.95

Due to U.S. Government Restrictions, this CD-ROM is only available for U.S. and Canadian customers.

Includes these books:

1. ***Applied Cryptology, Cryptographic Protocols, and Computer Security Models*** by Richard Demillo
2. ***Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition*** by Bruce Schneier
3. ***Contemporary Cryptology: The Science of Information Integrity*** edited by Gustavus J. Simmons
4. ***Cryptography and Data Security*** by Dorothy Denning
5. ***Cryptography: A New Dimension in Computer Data Security*** by Carl Meyer
6. ***Cryptography: Theory and Practice*** by Douglas Stinson
7. ***Handbook of Applied Cryptography*** by Paul C. Van Oorschot, Scott A. Vanstone, and Alfred Menezes

8-11 ***Military Cryptanalysis, Volumes I-IV*** by William Friedman

12. ***RSA Laboratories FAQ on Cryptography, RSA Laboratories Technical Reports, RSA Laboratories Security Bulletins, and CrytoBytes Newsletter***

12 Complete Books • Includes forward by Bruce Schneier

ORDER TODAY!
800-992-0549
U.S. & Canada

E-mail: orders@mfi.com
Fax Orders: 785-841-2624

Mail Orders
Dr. Dobb's CD-ROM Library
1601 West 23rd St., Suite 200
Lawrence, KS 66046-2703 USA

Dr. Dobb's CD-ROM LIBRARY

Go to our web site for more products from Dr. Dobb's CD-ROM Library:
www.ddj.com/cdrom/

Profile-Guided Optimizations

Optimizing object code using run-time information

Gary Carleton, Knud Kirkegaard, and David Sehr

Between 1992 and 1993, our compiler group at Intel was involved in developing optimizations for a new superscalar processor. We found these optimizations — instruction scheduling and code generation to avoid processor stalls — effective for many applications. However, we encountered some apps that didn't achieve the expected performance improvements. These programs seemed to have two common characteristics. First, they were large applications consisting of hundreds to thousands of functions, resulting in very large executable programs. Second, they spent little execution time in loops; instead, they actively exercised large portions of the executable code. Much of the application-execution time was spent in branches and function calls and returns.

One outgrowth of our experiences is our implementation of profile-guided optimizations. Profile-guided optimizations work by feeding information about how a program executes back to the compiler. This allows the compiler to focus its efforts more effectively on regions of programs that matter for execution time. In the case of the problem applications, profile information also enables the compiler to make the best use of the hardware resources available on the processor. First, it improves the performance of the instruction cache and paging mechanisms. It does this by helping the compiler to enhance program locality; that is, placing frequently executed code together, while moving infrequently executed code away. Second, profile information enables the compiler to improve the effectiveness of branch prediction hardware by rearranging branches.

The authors are engineers for Intel. They can be contacted at gary_carleton@ccm.sc.intel.com.

The profile-guided optimization techniques we developed have been incorporated in the Intel C/C++ Compiler Plug-in (http://developer.intel.com/drg/pentiumII/appnotes/compiler.htm), which can be integrated into Microsoft's Developer Studio. This makes it possible for you to use inline-assembly instructions that are currently not supported by Visual C++ 5.0. The Intel C/C++ Compiler Plug-in is compatible with Visual C++ 4.x or later compilers when it comes to command-line switches, inline-assembly format, object modules, library and DLL formats, debug and C++ symbol formats. Other optimizations provided by the plug-in (which are not currently available with Visual C++) include a rounding control option that optimizes floating point to integer conversions.

In this article, we discuss profile-guided optimizations — how the compiler collects run-time information — and present several of the optimizations performed by profile-guided compilers.

Figure 1 is a program fragment that illustrates a number of optimizations discussed here. The fragment contains a *while* loop enclosing an *if* statement. The shaded regions in the code represent basic blocks, the representation used by profiling. A basic block is a sequence of code in which either all or none of the code executes. Usually, a basic block ends with a branch, call, or return. Figure 2 illustrates the control-flow graph built from those basic blocks. Edges in the graph show how the flow of control can transfer between basic blocks, and each edge is labeled with the number of times control transfers through it.

Give the Compiler More Information

Most compilers make optimization decisions based on static heuristics. Notably, a compiler may decide whether to inline a function based on either its size or whether it is called from within a loop. Profile guidance enables the compiler to make better optimization decisions by using knowledge about how the program actually runs. With this profile information, the compiler can do a better job of inlining functions, ordering basic blocks, allocating registers, and so on.

Three-Step Build Process

In contrast to a traditional compiler that uses a single compilation to optimize an application, profile-guided compilation requires three phases:

- The first phase is the instrumented compilation. Instrumented compilation typically produces an executable program that contains probes in each of the basic blocks of the program. Each probe counts the number of times a basic block executes. If the block is a branch, the probe records the direction taken by that branch.
- The second phase is the profiled execution. When run, the instrumented program generates a data file that contains the execution counts for the specific run of the program.
- In the third phase, information from the profiled execution (or executions) of the program is fed back to the compiler. This data is added to the compiler's control flow graph.

Basic Block Ordering

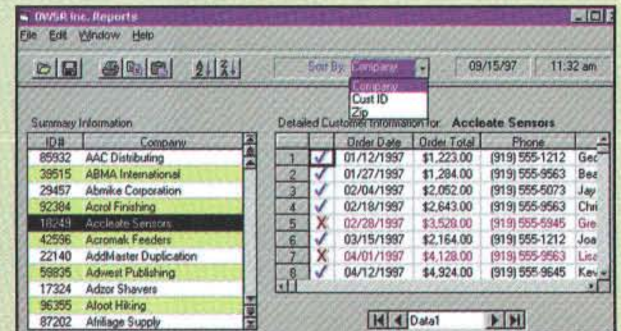
Without profile feedback, the basic blocks of a function are usually ordered simply to eliminate branches to other branches. The blocks in an *if-then-else* construct, for example, may be ordered according to some heuristic to try to improve branch-predictor effectiveness, but, in general, blocks are placed according to simple syntactic rules. For example, Figure 3(a) shows how a conventional compiler might

FarPoint's SPREAD

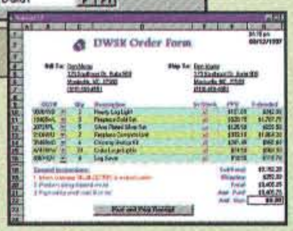
Using Spread, you and your team can quickly create powerful database front-ends, easily manage the display and entry of up to two billion items, print flexible reports, perform complex calculations, read and write files, sort data, or simply take advantage of its unsurpassed cell-level formatting.

POWER *flexibility* *SPEED*

All this from one control. Unmatched power, flexibility, and speed combine to make Spread the perfect complement to any application.



DATA BINDING



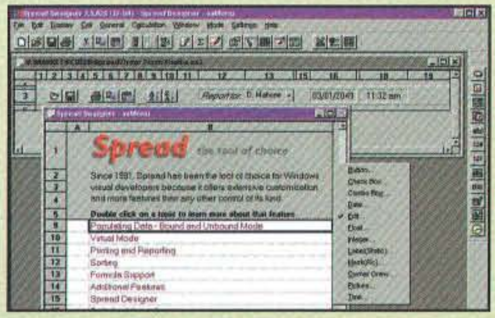
Easily insert unbound columns for greater flexibility ● Rearrange the order of the database fields or load only specified fields ● Automatically save any user-entered changes back to the database or validate any data changes ● Increase the data access speed using the Virtual Data Manager to control how many records are loaded at a time

SORTING & CALCULATIONS

Sort any range of data in ascending or descending order ● Perform any number of complex mathematical functions using the calc engine

GRID/SPREADSHEET

Provide two billion rows by two billion columns ● Drag and drop data anywhere in the control ● Lock cells to prevent text from being edited ● Create headers by spanning text across multiple cells ● Support the familiar clipboard copy/cut/paste commands ● Use the unique WYSIWYG Spread Designer to simplify programming tasks by allowing easy access to its rich feature set at design time



CELL FORMATTING

Provide the most popular data and database formats using 12 cell types ● Highlight any cell or block of cells using six border styles ● Customize your interface by changing the font and foreground and background colors for any cell ● Display up to 32K of data in each cell

PRINTING

Print comprehensive reports using any range of data ● Take advantage of Spread's Smart Print feature to reconfigure data to fit on a single page when possible

FarPoint technologies

Order today by calling 800-645-5913 or download a trial version at WWW.FPOINT.COM

FARPOINT TECHNOLOGIES INC.

Main 919-460-4551 *Sales* 800-645-5913 x3021 fpsales@fpoint.com

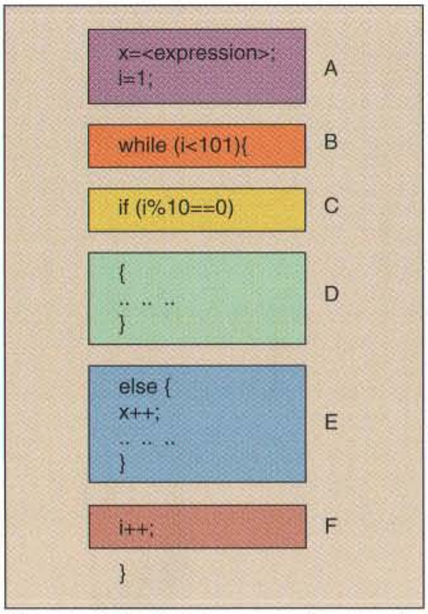


Figure 1: *Sample program fragment.*

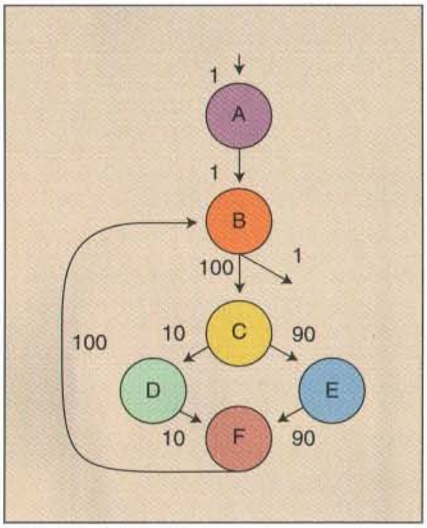


Figure 2: *Control-flow graph.*

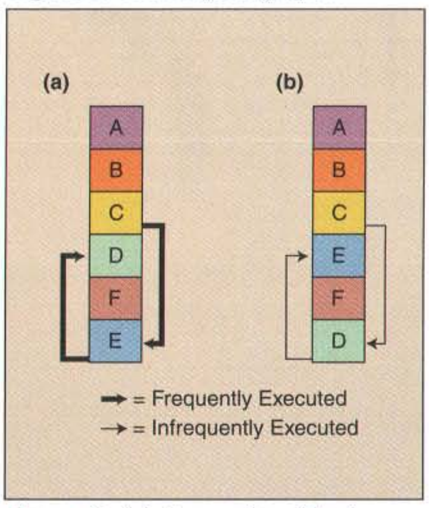


Figure 3: *(a) Conventional basic block ordering; (b) improved basic block ordering.*

(continued from page 98)

order the basic blocks in the executable for the example in Figure 1. The edges outside the layout show control transfers between blocks that are not adjacent. Basic block ordering rules often fail if the condition of an *if* statement is poorly predicted or predominantly takes the direction opposite of the heuristic. This results in three performance penalties that are addressed by profile-guided block layout.

First, many processors employ a first-time branch-prediction heuristic (for instance, "forward branches not taken, backward taken"). If a branch in a program does not conform well to this heuristic, then it is possible that the first-time prediction may be wrong most of the time. In the layout given in Figure 3(a), this prediction will be wrong because the first iteration of the loop exercises the code in block E rather than block D. This is relatively unimportant when the program is small and the branch executes a large number of times.

If the program is small enough, the processor can remember a prediction for each branch in the program, using the branch's history to predict its future behavior. However, for large applications, this may not be possible since there may be too many branches for the branch predictor to remember. If this happens, then static predictions will again be used for the branch. Using profile-guided block ordering, a compiler is able to select branch conditions to make better use of static branch prediction heuristics. Figure 3(b) shows a layout that makes better use of static prediction.

Second, compilers usually lay out blocks according to a static heuristic to reduce the number of branches along some path, say the *then* part of an *if-then-else*; path C-D-F in Figure 3(a). This choice requires an additional jump in the *else* part (block E). If the *else* part is executed more frequently, more jump instructions are executed. From the profiling information, the jump will be executed 90 times if the blocks are laid out as in Figure 3(a). A profile-guided compiler attempts to place blocks to minimize the number of jumps along the most frequently executed path. With profiling information, the compiler will note that the path from D to F executes less frequently than the path from E to F, and will place the code as in Figure 3(b). This results in an unconditional jump at the end of block D that executes only ten times.

Third, some blocks of a function—often error handling and recovery code—are seldom (or never) executed. Compilers often place such seldom-executed blocks in the middle of a function, filling the instruction cache and virtual memory paging system with code that is never executed. This is probably not significant for small programs, but for large applications, it's better to move infrequently executed code out of line to reduce the amount of code that is paged in or fetched into instruction cache. Profile-guided compilers can determine that such code is infrequently executed and move blocks out of the main body of the function, typically to the end of the function. This results in less instruction cache pollution and, to a lesser extent, fewer pages brought in during a run.

Profile-guided compilers may also be capable of moving infrequently executed blocks out of the code containing the function, and placing them at the end of the entire application. This function splitting can result in significant improvements in paging behavior, while retaining the functionality contained in the infrequently executed code. This optimization has a number of interesting implications in compiler design, such as how to compute address sizes.

Register Allocation

One of the most powerful optimizations performed by a compiler is register allocation. A register allocator attempts to assign registers to variables in a way that minimizes the number of times variables are loaded from memory. Because it is common to have more variables than available processor registers, a register allocator must decide which variables to keep in registers, in which portions of a function, and which variables to reference from memory.

Typically, the register allocator uses a static heuristic, preferring to keep variables used in loops in registers, for example. This results in register allocation that is locally optimized within the loop, but it may not produce the best total performance.

For example, in Figure 1, if block D contains a computation that uses a large number of registers, then the compiler may not keep x in a register. If there were variables that were live across the loop, the compiler might move register stores to memory (spills) and reloads outside the loop. If the loop has few iterations, this decision may be worse than placing the spills and reloads inside the loop. Furthermore, in the example, the loads and stores required to perform the increment in block E may be expensive. Hence, keeping x in a register, spilling it on entry to block D, and restoring on exit from that block results in much better code and ten loads and stores of x rather than 90. Profile-guided compilers are able to use execution frequency to select points for spills and reloads that are infrequently executed. This allows the register allocator to use registers for variables in regions that are heavily executed, and avoid spilling variables around infrequently executed regions.

Function Inlining

Function inlining is another optimization that benefits from profiling. Function inlining substitutes the code associated with a procedure call (call site) with the actual code for the function called (callee). Inlining removes the call overhead (setting up parameters, and so on) and makes it possible to optimize the callee in the context in which it is called.

Profile information provides execution counts of the call sites, thus enabling the function inliner to focus on the important call sites within the program. Function inlining without profile information may inline functions at call sites that are not important for performance, thereby increasing code size unnecessarily. It is important to control this code-size growth when inlining because it may affect instruction cache misses.

It is possible to profile before inlining, and vice versa. Each approach has its advantages. For example, the program in Example 1 illustrates a case where inlining before profiling is advantageous. In Example 1, if profiling occurs before inlining, then the profiling information for *InlinedProc* will reflect its behavior over all sites where it is called. Suppose a condition usually evaluates to False. Profile-guided optimization would arrange the branch so that the False block comes before the True block. However, if a condition is usually True when *InlinedProc* is called from A, then this results in exactly the wrong optimization for a copy of *InlinedProc* inlined into A.

```
void A (...) {
        InlinedProc (...);
   ...
}
void InlinedProc (...) {
        if (condition)
             // Code that is frequently executed when called by A
        else
             // Code frequently executed when called by others
}
```

Example 1: *A case where inlining before profiling is advantageous.*

On the other hand, Example 2 illustrates that it is difficult to do inlining before profiling, because recursive functions cannot be completely inlined. Furthermore, inlining all function calls, even in nonrecursive functions, may result in an executable program so large that it is not practical to run.

In our compiler implementation, we profile before making any inline decisions. To date, we have not observed cases where the behavior of a function differs dramatically based on its caller, which seems to support this choice.

Function Layout

Function layout is a form of code placement similar to basic block reordering. Function layout is not intended to reduce inaccurate branch predictions, but to improve instruction paging and instruction cache misses. This is achieved by placing functions with frequent calls between them close to each other in the executable code, making it more likely that the functions will be on the same page or on pages that can be kept in main memory simultaneously. For small functions, this also helps keep the functions in the instruction cache simultaneously.

A call graph shows the calling relationships between functions. There is one node for each function, and a line between a function A and a function B if A calls B. The edges have a weight that is

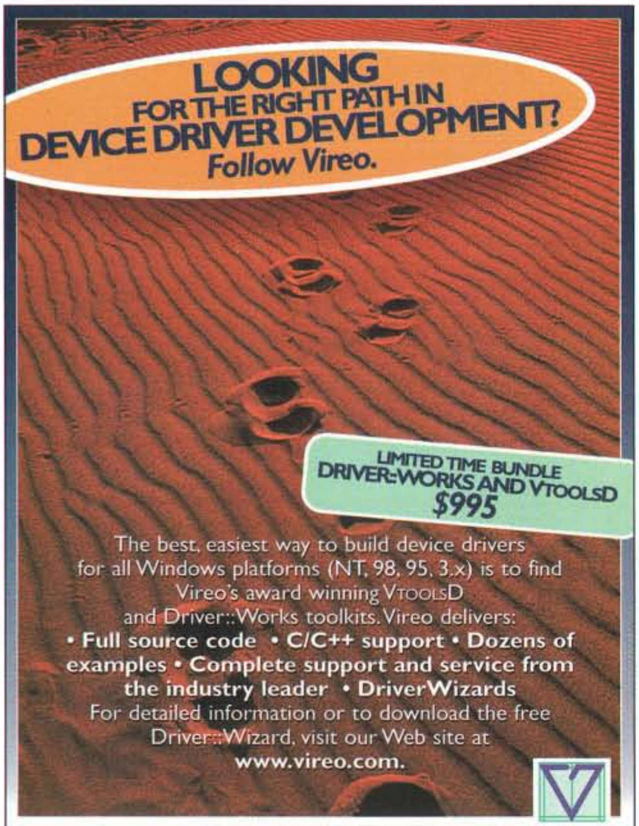
Dinkumware®, Ltd.
Genuine Software
www.dinkumware.com

Let Dinkumware help you meet the challenges of the new C++ Standard.

- The **Dinkum C++ Library** now conforms to the recently adopted ANSI/ISO C++ Standard.
- The **Dinkum® Abridged Library** provides the most used subset, including STL, for building leaner and faster applications.
- The **Dinkum EC++ Library** lets you code without templates, exceptions, or other space-eating language features.
- The **Dinkum EC++ Proofer™** ensures that your C and EC++ libraries remain true to their specification.
- **Dinkum Customization Services** bring your C Library up to date, and ready for the new C++ Standard as well.

Phone 1 888 4 DINKUM
Email sales@dinkumware.com
398 Main Street, Concord, MA 01742

AD LINK 348



30 Monument Square, Suite 135
Concord, MA 01742 USA
Phone: 978-369-3380 Fax: 978-318-6946 Email: sales@vireo.com
Vireo Software

AD LINK 416

the number of times A calls B in the instrumented run. Figure 4 shows a call graph that has the edges annotated with the call weights.

Function layout is a complicated problem and any reasonably fast solution to the problem is based on a heuristic. In our compiler, we have implemented three different algorithms for function layout. Two of them are based on a depth-first walk of the call graph, and the other is based on Pettis and Hansen's graph-collapse method (see "Profile Guided Code Positioning," by Karl Pettis and Robert C. Hansen, *Proceedings of the ACM SIGPLAN '90*).

Our descriptions of the three function layout algorithms use the call graph in Figure 4. Normally, the compiler places functions in the object file in lexical order, that is, the order in which they appear in the source file. Figure 5 shows the source order for our example.

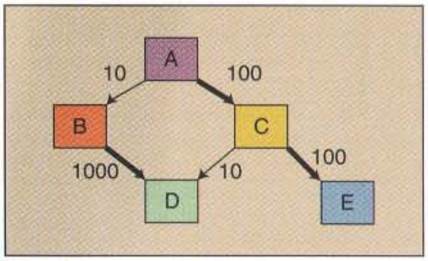


Figure 4: *Call graph.*

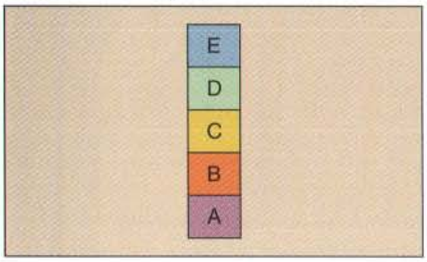


Figure 5: *Source order.*

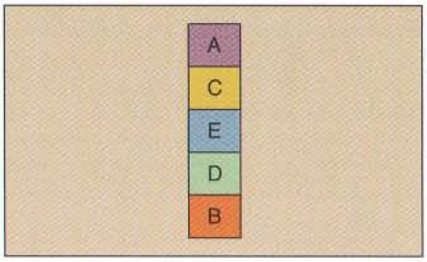


Figure 6: *DFO order.*

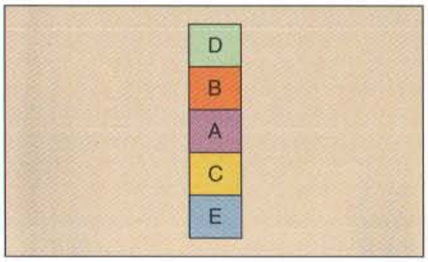


Figure 7: *HCC order.*

The simplest function layout method is called depth-first ordering (DFO). Starting at the top of the call graph, the functions are ordered by doing a depth-first walk of the call graph. If profiling information is available, the outgoing edges from a function are walked from most to least frequent. If no profiling information is available, the outgoing edges are walked in random order. Even without profile information, this does a better job in most cases than just ordering the functions in source order.

Figure 6 is the DFO layout of our call graph in Figure 4. Starting at the root, function A is first placed. Following the highest call weight, function C is placed next. Continuing down the call graph, E and then D are placed. The DFO walk then returns to A, then places function B. The layout is now complete since function D has already been placed.

The second method is highest-count caller (HCC). This method is also based

```
void B (...) {
      B(...)
}
```

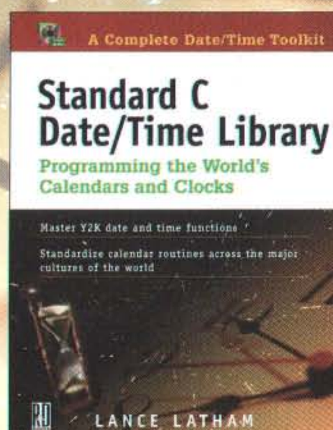
Example 2: *It is difficult to do inlining before profiling.*

Beat the Clock!

Standard C Date/Time Library

Programming the World's Calendars and Clocks

By Lance Latham



Does the year 2000 have you sweating late-night code? Use our complete library of C programming functions to master 'Y2K,' time on the net, ISO 8601, time stamp compression, or any other time/date application you encounter.

Using the astronomers' Julian Day calendar as a common denominator between the world calendars, Lance Latham has solved the conversion of virtually all (including the Chinese!) the world calendars. Need to convert dates in the Maya calendar to equivalent dates in the Egyptian calendar? Plug in the modules, write a few lines of code, and away you go! This toolkit provides all of the basic Western calendar/clock applications – written in clear, straightforward C – designed to fit into your application with a minimum of fuss. 576pp, CD-ROM included, ISBN 0-87930-496-0 **RD2735 $49.95**

Find R&D Books in your local bookstore
or call 800-441-4881 / Fax 785-841-2624 / e-mail: orders@mfi.com

www.rdbooks.com

on a depth-first walk of the dynamic call graph, starting at the top. However, before the walk, only the incoming call edge with the highest execution count is kept for each function. This way, a function will be placed closest to the function that calls it most frequently. Figure 7 shows the resulting function layout.

The last method is closest is best (CIB). The idea is to always pick the call edge in the dynamic call graph with the highest execution count remaining in the call graph. If there are two edges with the same weight, one is chosen at random. For each edge selected, the caller and callee nodes are merged into one node, and the call edges are updated accordingly. Two merged function nodes in the call graph indicate that the functions are placed next to each other. This process continues until the call graph is reduced to one node. When a node is merged into an already merged node, the function is placed closest to the function with the most immediate call relation. Figure 8 shows the result of CIB on our call graph.

Using function layout, we have measured up to 80 percent improvement in Instruction Translation Lookaside Buffer misses (ITLB is an on-chip cache that saves paging information to increase paging performance). On small programs, however, this may not mean a large performance improvement, since small applications normally are not limited by paging and instruction cache performance. On very large applications however, these effects may result in a measurable performance impact. On some larger applications, we have seen two to four percent improvements in instruction cache misses.

Conclusion

Profile-guided optimizations provide numerous opportunities to improve performance, only some of which have been discussed here. For instance, to improve the performance of C++ programs, profile-guided optimizations must consider indirect function calls. C++ virtual function invocation can be optimized significantly by profiling not just the frequency of calls but also their targets. This enables the simple optimization of replacing the indirect function invocation by an *if* test for the predominant type, using the indirect call only when that test fails. Currently, our compiler does not profile the targets of indirect function calls, but we consider this the most significant extension yet to be added. In addition to reducing the number of indirect calls, this optimization also enables better function placement, better function inlining, and better compiler analysis.

One of the challenges for profile-guided optimizations is to achieve wide acceptance and usage by software vendors. The more-complicated build process required by application developers is often an impediment to wide acceptance. Furthermore, the profiled run of the program typically takes significantly longer than an optimized run, and the application developer may not know what constitutes a representative data set. The benefits of the additional overhead in first doing an instrumented compilation, then running the application with representative data input sets, and finally compiling with feedback of the profile information, must be understood and appreciated before profile-guided optimizations will become a part of the normal application build process. However, for large applications that are executed frequently, profile-guided optimizations may prove to be a significant source of performance.

DDJ

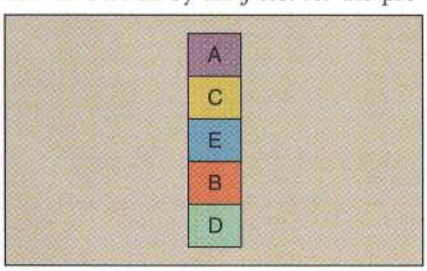


Figure 8: *CIB order.*

The longest continuously advertised C product in the history of ... mankind.

PC-lint

for C/C++
Version 7.5
presents Bug # 604

```
#include <iostream.h>

int &var_selector(int n)
    {
    int var_1 = 1;
    int var_2 = 2;

    if( n == 1 ) return var_1;
    else return var_2;
    }

int main()
    {
    int &x = var_selector(1);
    cout << "value = ";
    cout << x;
    return 1;
    }
```

Instead of a value of "1" *some strange number was printed. What went wrong? (N.B. some compilers, but not all, catch this one) Call if you need a hint or visit our web site at* www.gimpel.com

PC-lint for C/C++ will catch this and many other bugs. It will analyze a mixed suite of C and C++ modules to uncover bugs, glitches, quirks and inconsistencies.

Version 7.0 introduced several spectacular and revolutionary innovations in the art of program validation. Taking clues from initializers, assignments, and conditionals, variable and member values are tracked, enabling reports on potential uses of null pointers and out-of-bounds subscripts. Almost 100 standard functions are rigorously checked.

Version 7.5 pushes the envelope of lint detection still further. User-defined semantic checking of functions offers a unique language to express argument constraints, inter-argument relationships, and return values. There are 75 new messages including cradle-to-grave checking of pointers and the latest admonitions and perorations of Scott Meyers.

Plus Our Traditional C/C++ Warnings: Uninitialized variables, inherited non-virtual destructors, strong type mismatches, ill-formed macros, inadvertent name-hiding, suspicious expressions, etc., etc.

Full C++ Support - PC-lint for C/C++ is based on the ARM and is tracking the latest ANSI/ISO draft. It supports both Borland and Microsoft C/C++.

PC-lint for C/C++ *$239*

Numerous compilers/ libraries supported. Runs on Windows 95, Windows NT, MS-DOS (built-in DOS extender), and OS/2. 30 day money back guarantee.

FlexeLint for C/C++

The same great product for other operating systems. Runs on all Unix systems, VMS, mainframes, etc. Distributed in shrouded C source form. Call for pricing.

Gimpel Software

3207 Hogarth Lane, Collegeville, PA 19426

CALL TODAY (610) 584-4261 Or FAX (610) 584-4266

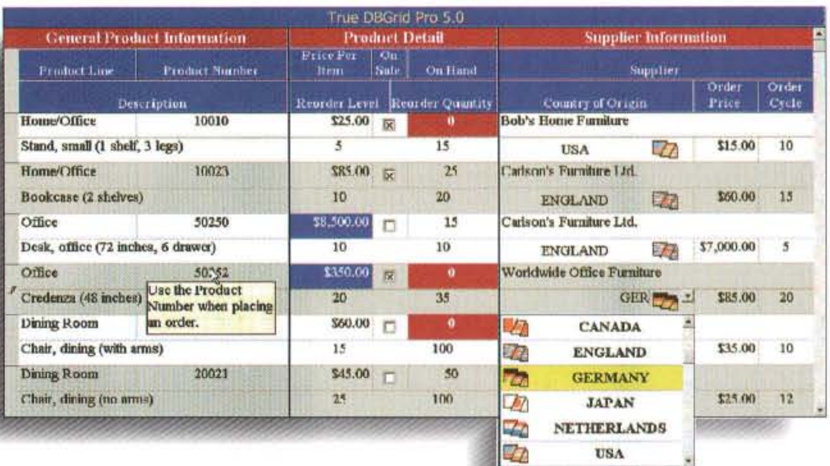
www.gimpel.com

PA add 6% sales tax.

PC-lint and FlexeLint are trademarks of Gimpel Software

The Official Upgrade Grid for Microsoft® Visual Basic® and Visual C++®

Single OCX for VB and VC++



Used by Developers in over 400 of the Fortune 500 Companies!

- Flexible Data Modes: 1 Bound, 3 Unbound, and 1 Array-based Storage Mode (no data access code required)
- Multiple Lines per Record
- Word and Excel-like Styles
- Alternating Row Styles
- Context-sensitive Row and Cell Styles
- In-Cell Graphics, Radio Buttons, Command Buttons, Check Boxes, and Drop-down Combo Boxes
- Data-aware Drop-down List Box
- Excel-like Splits and Split Headers
- Display and Input Masking
- Context-sensitive CellTips
- Multiple Row/Column Selection
- Drag and Drop Cell Contents
- Supports Unbound Columns
- Drop-down Multiline Text Editor
- Reusable Grid Layouts
- Automatic Data Translation for Display
- Interactive WYSIWYG Design-Time Editor
- Free Run-Time Distribution
- The Most Responsive Technical Support in the Industry
- and much more...

Upgrade to the best...

TRUE DB GRID® PRO 5.0

The Fastest Database Grid for VB!

Includes support for VB4 and VB5 plus:

- 16-bit and 32-bit OCXs
- Thorough 400 Page Manual and On-line Help
- Add-in Grid Design Assistant
- Add-in Grid Migration Utility
- 18 VB Tutorial Projects
- 5 VB Sample Projects

$249.95

*Call for Upgrade pricing

NEW

TRUE DB GRID® PRO 5.0 VC++

The Easiest-to-use ActiveX Grid for VC++!

Includes full support for VB plus:

- Support for 32-bit OCX in VC++ 5.0 using the MFC Class Wizard
- Access Data using the MFC DAO Classes, RDO Classes, or your own Custom Data Source
- True DBGrid Pro 5.0 VC++ Object Class Library
- 450 Page Manual and On-line Help Written Specifically for VC++ Developers
- One year of Technical Support
- One year of Free Upgrades
- 10 VC++ Tutorial Projects
- 10 VC++ Sample Projects

$399.95

*Call for Upgrade pricing

To Order

800.858.2739 (sales)
412.681.4343 (voice)
412.681.4384 (fax)
sales@apexsc.com (e-mail)

APEX Software Corporation
4516 Henry Street
Pittsburgh, PA 15213

Download *FREE* True DBGrid Pro 5.0 Evaluation Version Today
www.apexsc.com

All products and brand names are trademarks and/or registered trademarks of their respective holders.

AD LINK 422

Customizing DDX/DDV

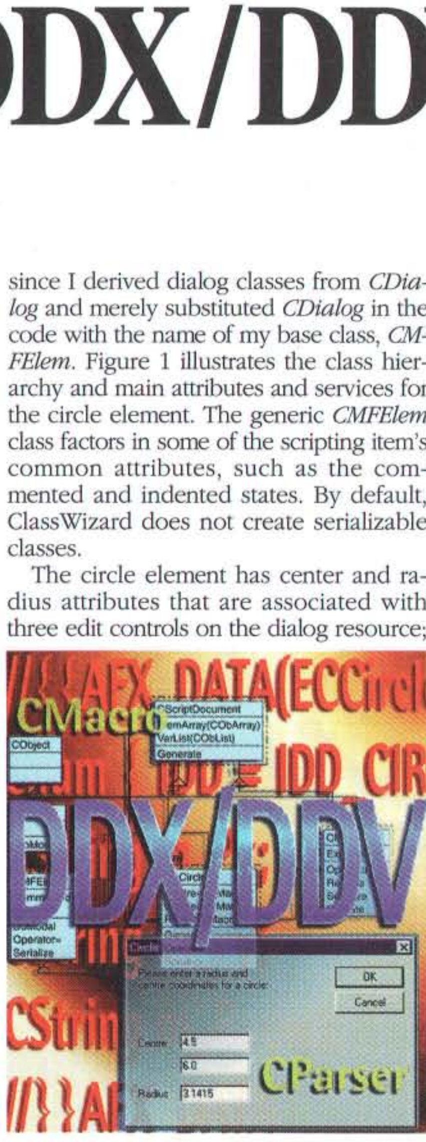
Use this trick with finite sets of scripting elements

Jean-Denis Bertron

I recently had to create a system that would let users put scripts together by picking programmatic elements and placing them in a list. I decided to provide users with an interface similar to that of installation systems, as a front end to a CGM generator library. (CGM, short for "Computer Graphics Metafile," is an ISO standard for 2D computer graphics storage and exchange of images. If you are familiar with the TIFF, WMF, or VRML formats, you know what I'm talking about.) The technique I developed uses custom data exchange routines to add macro processing to Windows dialog boxes. You can use this interface with installation systems, serial-communication packages, or just about any environment where the set of scripting elements is finite.

Since all scripting elements require parameters, I decided to create a C++ class derived from *CDialog* for each. By associating a dialog resource to each class, this design eliminates the need to store parameter data elsewhere. Thus, the class for a typical element defining a circle is derived from *CMFElem*, a generic scripting item class, which is itself derived from *CDialog*. ClassWizard does not, however, support custom classes when creating a class for a new dialog resource. Nevertheless, this was a minor inconvenience, since I derived dialog classes from *CDialog* and merely substituted *CDialog* in the code with the name of my base class, *CMFElem*. Figure 1 illustrates the class hierarchy and main attributes and services for the circle element. The generic *CMFElem* class factors in some of the scripting item's common attributes, such as the commented and indented states. By default, ClassWizard does not create serializable classes.

The circle element has center and radius attributes that are associated with three edit controls on the dialog resource;



see Figure 2. ClassWizard generated Listing One (listings begin on page 137) for the parameters in this class after I used it to create data members for the controls.

In CGM, parameters such as coordinates can be integers or reals. This is why I chose *CString* for the parameter types. However, storing only constant data in the elements was too restrictive. Users would ask to change the value of parameters at run time to generate, for example, concentric circles. As it turns out, one solution to this problem is to create custom Dialog Data Exchange and Validation (DDX and DDV, respectively) functions. DDX exchanges data between dialog controls and their corresponding member variables in a dialog class; DDV validates this data.

Creating Custom DDX/DDV Functions

Microsoft technical note TN026 describes how to create custom DDX/DDV functions and add ClassWizard support for data types using the routines. The MFC samples include a program called "CHKBOOK" to illustrate the technique. Using this method, you can associate a custom type to any control on a dialog resource. You must first create a new data type. Next, you must create an exchange function to access the control's attribute and fill the data member. When data is going the other way, the function will update the control's state using the data member. Last, you must add a few lines to ClassWizard's configuration file in your project directory or in Visual C++'s directory. You can also implement a DDV routine for data validation, but this is often inadequate because the function must take exactly one additional parameter.

I chose to implement a class to store an expression as a string, validate, and resolve it when necessary. The classes *CMacro* and *CParser* (available electronically; see "Resource Center," page 3) do this. *CParser* implements a top-down parser I prototyped using Leopurd (see "Building Parsers with Leopurd," by Thor Mirchandani, *DDJ*, March 1996). *CMacro*, a storage class similar to *CString*, offers two main services: *CMacro* can parse or resolve the expression it stores. The class uses a *CParser* object to perform these operations. I chose to separate the expression storage from the parser for convenience because the parser's grammar would evolve. You could combine both

Jean-Denis is a software engineer for Henderson Software. He can be reached via e-mail at jd@cgm.com.

classes into one, but consider the impact on serialization: Will a small change, such as adding a flag, to the parser code affect the macro object's serialization?

Copying the expression from the macro to the parser's internal buffer causes some overhead and extra memory allocation. The grammar I used is simple, providing basic mathematical operators; see Example 1. It is fragmented because Leopurd cannot process left-recursive productions. Variables are prefixed with an ampersand when referenced. For instance, the parser will validate the expression *lg(&bits) +&sign*, where *&bits* and *&sign* are references to global expressions. I will discuss these later. *CParser* shows the implementations of the DDX routine and the serialization and assignment operators associated with the *CMacro* type.

By adding the two lines in Example 2 in my project's CLW file under the [General Info] Section, I enabled ClassWizard's support for a *CMacro* type. I could now use a *CMacro* object to access an edit control, listbox, or combo box. For all existing classes with data members linked to controls in a resource, I had to remove the data member and add it back as a *CMacro*. For class *ECCircle*, the new data members coded by ClassWizard look like Listing Two. Because all my dialog-derived classes are serializable, I had already written code for *ECCircle* to store or load the data members as in Listing Three.

I needed to write code to serialize a *CMacro* object. To avoid changing this code, I decided to create the overloaded << and >> operators for the type. Both operators merely call the *CMacro Serialize* member function, which does all the work. These functions are located at the bottom of the listing for *CParser* (available electronically).

```
macro     :          expr EOI;
expr :    stmt Zexpr;
Zexpr     :          EMPTY ¦
          '+' expr ¦
          '-' expr ¦
          OR expr ¦
          XOR expr ;
stmt      :          term Zstmt;
Zstmt     :          EMPTY ¦
          '*' term ¦
          '/' term ¦
          '%' term ¦
          AND term ;
term      : number ¦
          '(' expr ')'  ¦
          LOG '(' expr ')'   ¦
          LN '(' expr ')'   ¦
          LG '('  expr ')'   ¦
          EXP '(' expr ')'   ¦
          ABS '(' expr ')'   ¦
          SIN '(' expr ')'   ¦
          COS '(' expr ')'   ¦
          TAN '(' expr ')'   ¦
          ASIN '(' expr ')'   ¦
          ACOS '(' expr ')'   ¦
          ATAN '(' expr ')'   ¦
          NOT '(' expr ')'   ¦
          DIV '(' expr ',' expr ')'
     ¦
          CNP '(' expr ',' expr ')'
     ¦
          POW '(' expr ',' expr ')'
     ¦
          ANP '(' expr ',' expr ')'
     ;
number    :          HEXVAL ¦
          afloat ¦
          '+' afloat ¦
          '-' afloat ¦
          NAME ;
afloat    : INTEGER decimal exponent
     ;
decimal   :          EMPTY ¦ '.' INTEGER ;
exponent  :          EMPTY ¦
          'E' Eexp;
Eexp      :          INTEGER ¦
          '-' INTEGER ¦
          '+' INTEGER ;
```

Example 1: *Grammar used to provide basic mathematical operators.*

Global Variables

To give expressions access to global variables, I had to add an extra parameter to the two resolution functions in *CMacro*. It is a pointer to the document, because in my application, the document is the place where elements are resolved. This pointer to the parent document is stored in the *CParser* class during the resolution. Clearly, this is preferable to passing the pointer to all functions involved in parsing, which would waste stack space, a precious resource in top-down parsers. It is safe to save a pointer to the parent document since it is a C++ object, not a window.

The validation routine does not need such a pointer, unless you want to validate the existence of a variable without resolving its value. I added the extra parameter to it, although I do not use the feature. The function *CParser::number_action()* uses the pointer to call the document's *FindByName* member to locate and evaluate a global variable.

I used a collection of *CMacro* types to store global variables at the document level. This has the advantage of allowing code reuse. Also, users only have to learn one macro grammar. In addition, *CMacro* is a great candidate for being stored in a *CObArray* collection, since its parent class is *CObject*. Finally, the resolution of global variables can stay consistent with the rest of the hierarchy. For example, a scripting element's expression can reference a global macro, which can reference any other global macros. This was an easy design choice, although other alternatives might have been equally interesting. For efficiency, you can consider storing global variables in a collection class using the C++ Standard Template Library.

To prevent reentrance of the code, as might happen with circular references between macros, I set an internal flag inside

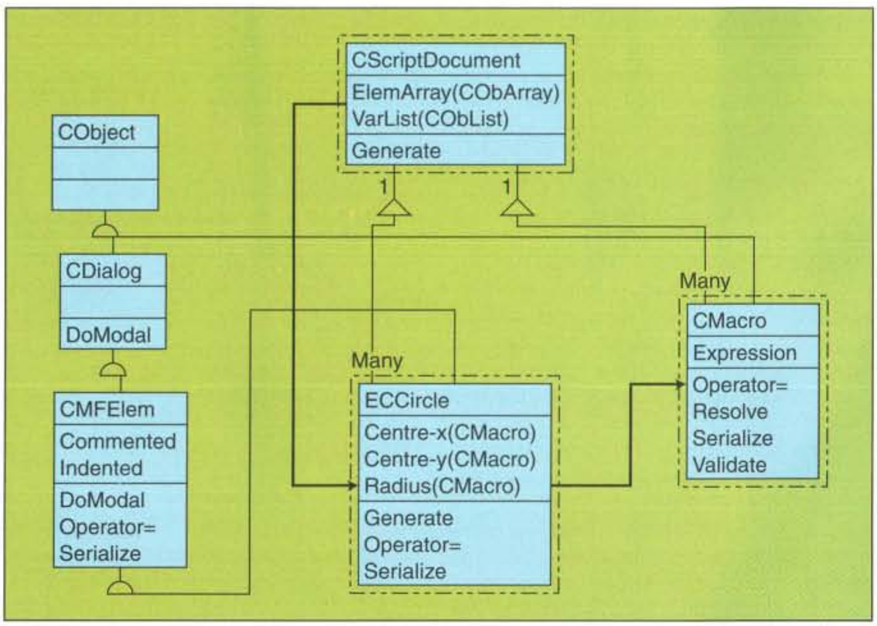


Figure 1: *Class hierarchy and main attributes and services for the circle element.*

```
ExtraDDXCount=1
ExtraDDX1=ELIMNn;;Value;CMacro;;Macro;Macro Expression
```

Example 2: *Adding these lines enables ClassWizard's support for a* CMacro *type.*

the *CMacro* object before it is resolved and the parser checks before attempting to resolve the macro.

CMacro has an assignment operator. Good C++ class design usually mandates that you define a constructor, destructor, copy constructor, and assignment operator. I wrote *CMacro*'s assignment operator to simplify the code in scripting elements' assignment operators, where *CMacro* types are being copied; see Listing Four. Note that the parameter to the operator is a reference to an object of the base class. Assignment operators are not inherited. The default memberwise assignment operator supplied by the compiler takes an *ECCircle* reference as a parameter. Why, then, is this virtual operator used when invoked via pointers to *ECCircle* elements? Because a standard conversion is available between an *ECCircle* object and an object of the base class, *CMFElem*. The operator is used as the default assignment by the compiler.

Why did I need virtual assignment operators for my scripting elements? The application uses the elegant Undo/Redo design pattern described by Jim Beveridge in "Implementing Multilevel Undo/Redo" (*DDJ*, February 1996). The Undo/Redo framework keeps track of user actions by saving copies of modified objects. The Undo/Redo internals are better isolated from the element's implementation by making copies of scripting elements using an assignment operator on element pointers. This explains why the operator takes a base class reference as its parameter.

Alternatives

I did not seriously consider alternatives to this method for implementing the macro functionality. You can certainly design a similar system around a few-high level classes, such as a "macro-expression manager" to process expressions that are submitted for evaluation. This architecture separates the expression parsing from dialog classes, eliminates the need for a macro type and preserves data members as *CString* (significant benefits when dialogs are created at run time or when portability is important). The Extra DDX feature is platform dependent and tool specific. However, because my application features over 200 dialog classes, relying on ClassWizard's support for the *CMacro* data type and the transparency of serialization and assignment operators was a time-saving strategy in its development.

I did not illustrate how to use a custom DDV function, primarily because the expression's validation occurs in the DDX routine. This shows that DDV routines are optional, although a DDX routine is necessary if you wish to use a DDV function.

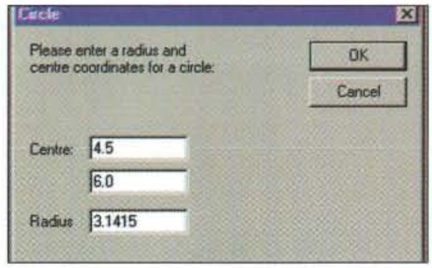


Figure 2: *The circle element has center and radius attributes which are associated with three edit controls on the dialog resource.*

Conclusion

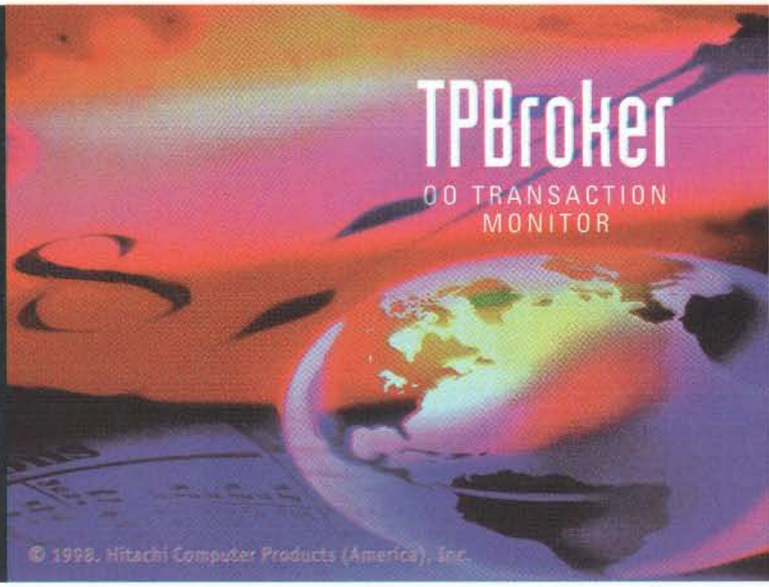
A "lite" version of my application is available electronically. It features only a few scripting elements. The grammar for the parser is included, and the Undo/Redo framework is functional. Credit goes to Thor Mirchandani and Jim Beveridge. I would appreciate hearing about the ways you have benefited from this article. Anyone have a self-translating *CEdit* field?

DDJ

(Listings begin on page 137.)

HOW LONG DOES IT TAKE TO GET THE ONLY OBJECT-ORIENTED TRANSACTION MONITOR FOR MISSION-CRITICAL APPLICATIONS?

HOW FAST IS YOUR NETWORK CONNECTION?



DOWNLOAD

AN EVALUATION

COPY AND

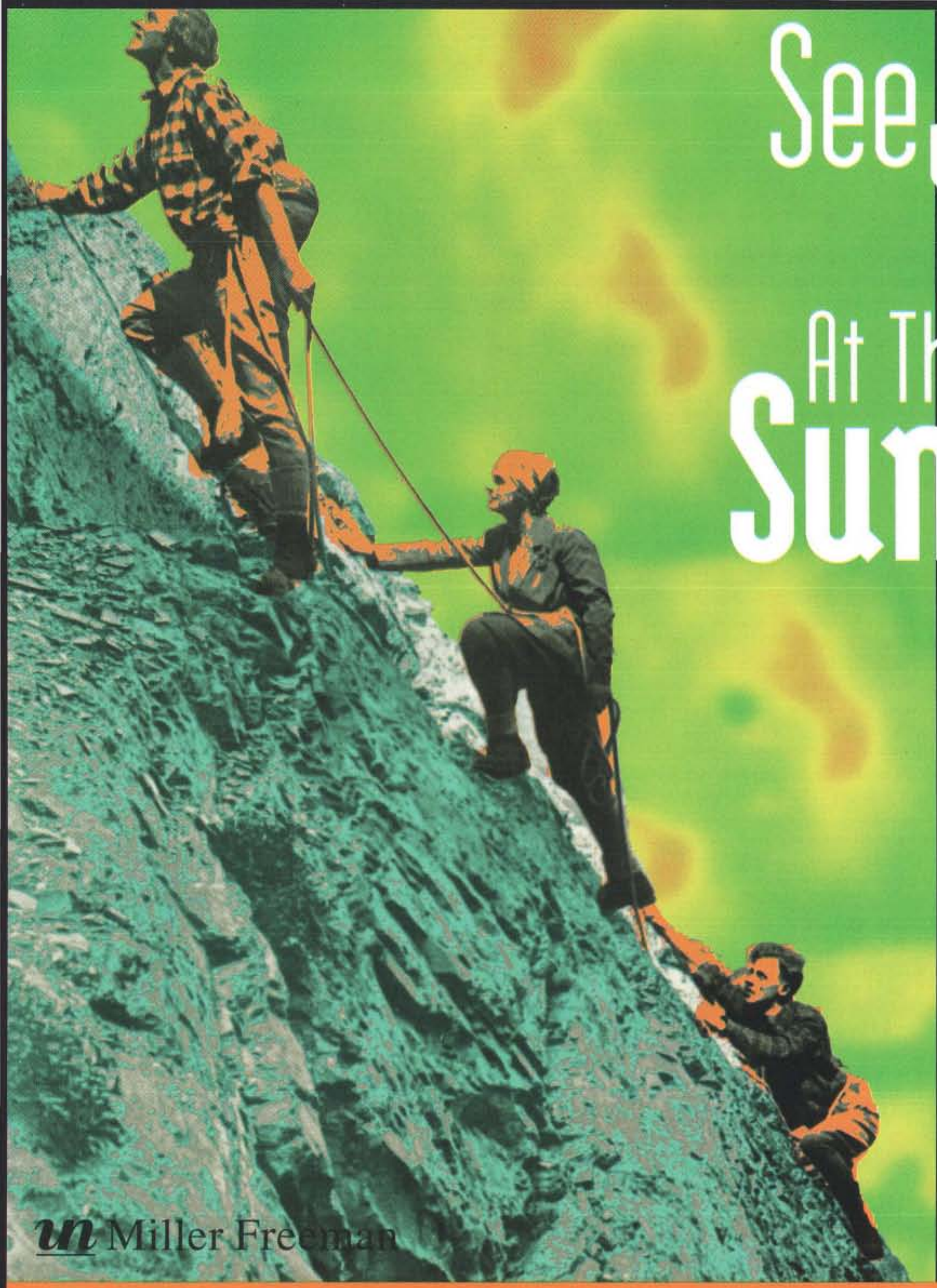
FIND OUT MORE:

WWW.TPBROKER.COM

To put OLTP functionality in a Web-based application, most people still think they have to develop their own solution. But all of a sudden, you know something they don't. TPBroker from Hitachi is here. Now! It's the only commercially-proven solution available. It can save you time and money. You should also know that it has excellent interoperability with your existing systems. And that it supports flat and nested transactions. It also features a high-performance, scalable, distributed OTS architecture. But most importantly, you should know our Web address: www.tpbroker.com and this number: 800.558.1413. So you can know even more things that most other people don't.

HITACHI SOFTWARE

AD LINK 374



Web Design & Development '9

6 focused tracks examine th trends and give you the lates tools and techniques:

- Strategy
- Content
- Design
- Usability
- Programming
- Back End

Being a Web Professional Requires Some Remarkable Skills

It takes your entire team working together. To stay on top, you need solid training from the best minds in the business. From the fundamentals of site usability to cutting-edge programming techniques, Web Design & Development '98 sets the standard for giving web professionals what they need to succeed. Web '98 provides practical training from the leading experts in information design; dynamic HTML vs. Java; XML; privacy; meta-data; browser hell and more. Visionary keynotes. Peer networking and brainstorming. And an exhibition of powerful tools from the hottest companies around. Join us for the most intense gathering of web brilliance the West Coast has even seen. Because when you're serious about your job, there's only one place to go.

MOSCONE CENTER
SAN FRANCISCO

CONFERENCE June 21-25 199

EXHIBITION June 22-24 199

WEB98
design & developmen
where web professionals converg

www.mfweb.com

Feeling nostalgic? Contact us by phone at 800-441-8826 or by e-mail at web98@mfi.com.

Y2K COMPRESSION

Listing One

```
/* routine:  sliding_window
*  Author:  Robert Moore
*  Takes a reference four digit year (year), a window width above the
*  reference year (upperwinwith), and a one or two-digit year being queried
*  (qyear), and returns a four-digit representation (ryear) of the query
*  year based on the window and the reference year if the query year is one
*  or two digits. Query years greater than two digits are just returned.
*  Setting reference year to a constant would make this a fixed window.
*/
int sliding_window(int year, int upperwinwidth, int qyear)
{
  int yeardate, centurydate, ryear, temp;
  ryear = -1;  /* status that will be returned on error */
  if (qyear < 100)  /* the query year is a two digit year --
                       convert to four */
    {               /* bounds check */
      if ((year >= 0) && (0 < upperwinwidth) && (0 <= qyear))
    {  /* everything a proper bound */
      yeardate = year % 100;  /* convert year to a year */
      centurydate = year - yeardate;  /* and century */
      temp = yeardate + upperwinwidth;  /* calculate upper window limit */
      if (temp < 100)
        {
          if (temp <= qyear )
        ryear = centurydate - 100 + qyear; /* year wraps to previous */
          else                              /* century:  case 1b */
        ryear = centurydate + qyear; /* same century:  case 1a or 1c*/
        }
      else
        {
          if (qyear < (temp - 100))
       ryear = centurydate + 100 + qyear; /* year in next century: case 2c */
          else
       ryear = centurydate + qyear; /* same century:  case 2a or 2b */
        }
    }
    }
  else
    ryear = qyear; /* three or more digit year--don't convert */
  return ryear;
}
```

Listing Two

```
/*--------------------------------------------------------------
/*  Authors:  D. Greg Foley, Robert L. Moore
/*  Description: This function determines if a four digit year is a leap
/*    year. A year is a leap year if it is divisible by 4 but not by 100,
/*    except years that are divisible by 400.
/*  Parameter Description:
/*  int Year - a four-digit year
/*  returns 1 if the year is a Leap Year
/*          0 if the year is NOT a Leap Year
/*  Notes: For clarity of the algorithm no error checking is included.
/*--------------------------------------------------------------*/
int LEAPYEAR(int Year)
{
    if ((Year % 4 == 0 && Year % 100 != 0) || Year % 400 == 0)
      return 1;   /* This is a LEAP year */
    else
      return 0;   /* This is NOT a LEAP year */
}
```

Y2K SOLUTIONS

Listing One

```
01  BILLCOM-RECORD.
    05  B-EFF-DATE.
        10  B-EFF-YY        PIC  9(02)        VALUE 0.
    05  B-CONV-DATE.
        10  B-CONV-YY       PIC  9(02)        VALUE 0.
    05  FILLER              PIC  9(96).
```

Listing Two

```
01  BILLCOM-RECORD.
    05  B-EFF-DATE.
        10  B-EFF-YY        PIC  9(04)        VALUE 0.
    05  B-CONV-DATE.
        10  B-CONV-YY       PIC  9(04)        VALUE 0.
    05  FILLER             PIC  9(96).
```

Listing Three

```
WORKING STORAGE SECTION.

01  POLICY-START-DATE.
    05  POLICY-START-DATE-YY     PIC 9(02) VALUE ZEROS.
    05  POLICY-START-DATE-MM     PIC 9(02) VALUE ZEROS.
    05  POLICY-START-DATE-DD     PIC 9(02) VALUE ZEROS.

01  WS-CONVERT-DATE.
    05  WS-CENTURY           PIC 9(02) VALUE ZEROS.
    05  WS-CONVERT-YY            PIC 9(02) VALUE ZEROS.
    05  WS-CONVERT-MM            PIC 9(02) VALUE ZEROS.
    05  WS-CONVERT-DD          PIC 9(02) VALUE ZEROS.

01  WS-PIVOT-DATE.
    05  WS-PIVOT-YY          PIC 9(02) VALUE 30.

PROCEDURE DIVISION.

MOVE POLICY-START-DATE-YY TO WS-CONVERT-DATE-YY
MOVE POLICY-START-DATE-MM TO WS-CONVERT-DATE-MM
MOVE POLICY-START-DATE-DD TO WS-CONVERT-DATE-DD
PERFORM VALIDATE-DATE-ROUTINE
   EVALUATE TRUE
       WHEN WINDOW-YES
            IF WS-CONVERT-YY  IS GREATER THAN  OR EQUAL TO
                 WS-PIVOT-YY
               MOVE '19'            TO  WS-CENTURY
            ELSE
               MOVE '20'            TO  WS-CENTURY
            END-IF
   END-EVALUATE
END-PERFORM.
```

Listing Four

```
VALIDATE-DATE-ROUTINE. (Performed by the Windowing routine)
EVALUATE TRUE
    WHEN  WS-VALIDATE-DATE EQUAL ZEROS-YYMMDD
          MOVE ZEROS-CC TO WS-VALIDATE-DATE-CENTURY
          SET WINDOW-INDICATOR TO FALSE
    WHEN WS-VALIDATE-DATE EQUAL NINES-YYMMDD
          MOVE NINES-CC TO WS-VALIDATE-DATE-CENTURY
          SET WINDOW-INDICATOR TO FALSE
END-EVALUATE.
```

Listing Five

```
01 WS-DATE.
    05 WS-YY PIC XX.
    05 WS-MM PIC XX.
    05 WS-DD PIC XX.
01 WS-Y2K.
    05 WW-YY-Y2K PIC X(4).
IF WS-YY < 75
        PERFORM 1000-PRT-RTN.
```

Listing Six

```
******IF WS-YY < 75
    If WS-YY-Y2K < 1975
    PERFORM 1000-PRT-RTN.
```

Listing Seven

```
*****************************************************************
* ORIGINAL LINE OF CODE:
*       IF B-CLM-STUS-CD = 'O' AND CLM-CLG-REOP-DT-HD (STATS)
*            < TRN-ISS-DT-DRCO NEXT SENTENCE ELSE
*** Y2K START ***
        IF B-CLM-STUS-CD = 'O'
           MOVE CLM-CLG-REOP-DT-HD (STATS) TO WS-JULIAN-WORK-DT-Y2K
           PERFORM JULIAN-DATE-RTN-Y2K THRU JULIAN-DATE-RTN-Y2K-EXIT
           MOVE WS-JULIAN-WORK-DT-Y2K TO CLM-CLG-REOP-DT-HD-Y2K
           MOVE TRN-ISS-DT-DRCO TO WS-JULIAN-WORK-DT-Y2K
           PERFORM JULIAN-DATE-RTN-Y2K THRU JULIAN-DATE-RTN-Y2K-EXIT
           MOVE WS-JULIAN-WORK-DT-Y2K TO TRN-ISS-DT-DRCO-Y2K.
         * note the period
       IF B-CLM-STUS-CD = 'O' AND CLM-CLG-REOP-DT-HD-Y2K
< TRN-ISS-DT-DRCO-Y2K NEXT SENTENCE ELSE
*** Y2K END ***
```

HDF

Listing One

```
#include "hdf.h"
#define WIDTH    5
#define HEIGHT   6
main(int argc, char *argv[])
{
    uint8 palette_data[768];
    intn i;
    /*  Initialize the image array  */
    static uint8 raster_data[HEIGHT][WIDTH] =
                { 1,  2,  3,  4,  5,
                  6,  7,  8,  9, 10,
                 11, 12, 13, 14, 15,
                 16, 17, 18, 19, 20,
                 21, 22, 23, 24, 25,
                 26, 27, 28, 29, 30 };

    /*  Initialize the palette to standard linear grayscale  */
    for (i=0; i<256; i++) {
    palette_data[i*3] = i;
    palette_data[i*3+1] = i;
    palette_data[i*3+2] = i;
    }
    /*  Associate the palette with the image  */
    DFR8setpalette(palette_data);
    /*  Write the 8-bit raster image to the file  */
    DFR8addimage("example.hdf", raster_data, WIDTH, HEIGHT, 0);
}
```

Listing Two

```
#include "hdf.h"
#include "mfhdf.h"
#define LENGTH  3
#define HEIGHT  2
#define WIDTH   5

 main(int argc, char *argv[])
{
    /*  Initialize the image array  */
    static float64 scien_data[LENGTH][HEIGHT][WIDTH] =
                { 1.,  2.,  3.,  4.,  5.,
                  6.,  7.,  8.,  9., 10.,
                 11., 12., 13., 14., 15.,
                 16., 17., 18., 19., 20.,
                 21., 22., 23., 24., 25.,
                 26., 27., 28., 29., 30. };
    int32 dims[3]           = {LENGTH, HEIGHT, WIDTH};
    int16 scale0[LENGTH]    = {2, 4, 6};
    int32 scale1[HEIGHT]    = {1234567, 2345678};
    float32 scale2[WIDTH] = {2.2, 4.4, 6.6, 8.8, 11.0};
    float64 avg             = 15.0;
    int32 start[3]          = {0, 0, 0};
    int32 fid, sdid, dimid0, dimid1, dimid2;
```

(continued on page 110)

(continued from page 109)

```
    /*  Open file and initialize SD interface  */
    fid = SDstart("example.hdf", DFACC_CREATE);
    /*  Create named data set  */
    sdid = SDcreate(fid, "Sample Data Set", DFNT_FLOAT64, 3, dims);
    /*  Set up dimension zero  */
    dimid0 = SDgetdimid(sdid, 0);
    SDsetdimname(dimid0, "Dimension 0");
    SDsetdimstrs(dimid0, "The zeroth dimension", "mm", "2d");
    SDsetdimscale(dimid0, LENGTH, DFNT_INT16, (VOIDP)scale0);
    /*  Set up dimension one  */
    dimid1 = SDgetdimid(sdid, 1);
    SDsetdimname(dimid1, "Dimension 1");
    SDsetdimstrs(dimid1, "The first dimension", "cm", "8d");
    SDsetdimscale(dimid1, HEIGHT, DFNT_INT32, (VOIDP)scale1);
    /*  Set up dimension two  */
    dimid2 = SDgetdimid(sdid, 2);
    SDsetdimname(dimid2, "Dimension 2");
    SDsetdimstrs(dimid2, "The second dimension", "m", "4.1f");
    SDsetdimscale(dimid2, WIDTH, DFNT_FLOAT32, (VOIDP)scale2);
    /*  Write the data array to the data set  */
    SDwritedata(sdid, start, NULL, dims, (char *)scien_data);
    /*  Add one local attribute  */
   SDsetattr(sdid, "Average", DFNT_FLOAT64, 1, (char *)&avg);
    /*  Add one global attribute  */
    SDsetattr(fid, "Date", DFNT_CHAR8, 9, "10/29/93");
    /*  Close the data set, file, and interface  */
    SDendaccess(sdid);
    SDend(fid);
}
```

Listing Three

```
#include <string.h>
#include "hdf.h"
#include "mfhdf.h"

#define MAXRANK   3
#define LENGTH    3
#define HEIGHT    2
#define WIDTH     5
#define DATESIZE  9

main(int argc, char *argv[])
{
    float64 scien_data[LENGTH][HEIGHT][WIDTH];
    int32 dims[MAXRANK];
    int16 scale0[LENGTH];
    int32 scale1[HEIGHT];
    float32 scale2[WIDTH];
    float64 avg;
    int32 start[MAXRANK] = {0, 0, 0};
    int32 fid, sdid, dimid;
    int32 i, index, rank, nattrs, ndatasets, nglobals;
    int32 nt, count, status;
    intn size;
    char name[80], date[80];

    /*  Open file and initialize SD interface  */
    fid = SDstart("example.hdf", DFACC_RDONLY);
    status = SDfileinfo(fid, &ndatasets, &nglobals);

    /*  Read global attribute  */
    if (nglobals == 1) {
        status = SDattrinfo(fid, 0, name, &nt, &size);
        if ((strcmp(name, "Date")) && (nt == DFNT_CHAR8) &&
            (size == DATESIZE))
            SDreadattr(fid, 0, date);
    }
    /*  Open first data set  */
    index = SDnametoindex(fid, "Sample Data Set");
    sdid = SDselect(fid, index);
    SDgetinfo(sdid, name, &rank, dims, &nt, &nattrs);
    /*  Read in data if everything looks okay  */
    if ((rank == MAXRANK) && (dims[0] == LENGTH) && (dims[1] == HEIGHT)
        && (dims[2] == WIDTH) && (nt == DFNT_FLOAT64))
        SDreaddata(sdid, start, NULL, dims, scien_data);
    /*  Read local attribute  */
    status = SDattrinfo(sdid, 0, name, &nt, &size);
    if ((strcmp(name, "Average")) && (nt == DFNT_FLOAT64) &&
        (size == 1))
        SDreadattr(sdid, 0, &avg);
    /*  Read dimensions  */
    dimid = SDgetdimid(sdid, 0);
    SDdiminfo(dimid, name, &count, &nt, &nattrs);
    if ((nt == DFNT_INT16) && (count == LENGTH))
        SDgetdimscale(dimid, scale0);
    dimid = SDgetdimid(sdid, 1);
    SDdiminfo(dimid, name, &count, &nt, &nattrs);
    if ((nt == DFNT_INT32) && (count == HEIGHT))
        SDgetdimscale(dimid, scale1);
    dimid = SDgetdimid(sdid, 2);
    SDdiminfo(dimid, name, &count, &nt, &nattrs);
    if ((nt == DFNT_FLOAT32) && (count == WIDTH))
        SDgetdimscale(dimid, scale2);
    /*  Close the data set, file, and interface  */
    SDendaccess(sdid);
    SDend(fid);
}
```

WINDOWS CE

Listing One

```
class CEQuery
{
    friend class CCeFindDBDatabaseIterater;
public:
    CEQuery();
    virtual ~CEQuery();

    void addField(CCeDBQueryProp *);
    void setSortField(CCeDBQueryProp *);
    int numFields(void);

    CCeDBQueryProp * testOn(WORD, WORD);
    bool match(CCeDBRecord*, bool &, int &);
    CCeDBQueryProp * sortField();
    CCeDBQueryProp * operator[](int fieldNo);

private:
    bool matchField(CCeDBProp*, int);

    CCeDBQueryProp ** _fields;
    CCeDBQueryProp * _sortField;

    int _numFields;
    int _maxFields;

    WORD _index;
    bool _range;
}
```

Listing Two

```
class  CCeDBQueryProp : public CCeDBProp
{
public:
    enum FieldOperation {
        Equal, LE, GE
    };
    void setOperation(FieldOperation);
    FieldOperation getOperation();
private:
    FieldOperation _operation;
}
```

Listing Three

```
class CCeFindDBDatabase   : public CCeDBDatabase
{
public:
    CCeFindDBDatabaseIterater * Find (CEQuery & );
    bool TestFind (CEQuery & );
private:
    CCeDBQueryProp * FindKey(CEQuery &);
    bool IsIndexed(CEPROPID, WORD *);
};
```

Listing Four

```
class CCeFindDBDatabaseIterater
{
public:
    CCeFindDBDatabaseIterater(CEQuery *, CCeDBQueryProp *, CCeDBDatabase *);
    ~CCeFindDBDatabaseIterater();
    CCeDBRecord * current();
    CCeDBRecord * first();      // resets search
    CCeDBRecord * next();
    void stats(int &, int &, bool &);
private:
    bool _start;
    bool _eof;
    int _hits;
    int _tests;

    CCeDBQueryProp * _primaryQuery;
    CCeDBDatabase * _myDb;
    CEQuery * _myQuery;
    WORD _keyOrder;
}
```

Listing Five

```
void CExampleDlg::OnInit()
{
    CCeDBDatabase newDB;
    CCeDBProp sorts[2];

    sorts[0].SetType(CCeDBProp::Type_String);
    sorts[0].SetIdent(1);
    sorts[0].SetSortFlags(CCeDBProp::Sort_Ascending );
    sorts[1].SetType(CCeDBProp::Type_Long);
    sorts[1].SetIdent(2);
    sorts[1].SetSortFlags(CCeDBProp::Sort_Descending );

    if (newDB.Create(TEXT("JCSExample"), 123, 2, sorts) == NULL)
    {
        newDB.Open(TEXT("JCSExample"));
        newDB.Delete();
        newDB.Create(TEXT("JCSExample"), 123, 2, sorts);
    }
    newDB.Open(TEXT("JCSExample"));

    CCeDBRecord r;
    r.AddProp(&sorts[0]);
    r.AddProp(&sorts[1]);
    r.AddProp(new CCeDBProp(CCeDBProp::Type_Filetime, 3));
    for (int i = 0; i < 1000; i++)
    {
        TCHAR string_field[256];
        FILETIME filetime_field;
        SYSTEMTIME systemTime;
        int j;
        // generate three random values for fields
        CCeDBProp* p = r.GetPropFromIdent(1);
        string_field[0] = (TCHAR)('A'+Random()%26);
        for (j=1; j < (int)(5 + (Random()%10)); j++)
            string_field[j] = (TCHAR)('a'+Random()%26);
        string_field[j] = 0;

        p->SetString(string_field);
        p = r.GetPropFromIdent(2);
        p->SetLong(Random());
        p = r.GetPropFromIdent(3);

        memset(&systemTime, 0, sizeof(systemTime));
        systemTime.wYear = (unsigned short) (1900 + Random()%200);
        systemTime.wMonth = (unsigned short) (1 + Random()%12);
        systemTime.wDay = (unsigned short) (1 + Random()%28);
        SystemTimeToFileTime(&systemTime, &filetime_field);
        p->SetFiletime(filetime_field);
        newDB.AddRecord(&r);
    }
}
```

Listing Six

```
CCeDBQueryProp::FieldOperation _ops[] = {
    CCeDBQueryProp::LE, CCeDBQueryProp::GE, CCeDBQueryProp::Equal};
CCeDBQueryProp * qp;
CEQuery q;
if (m_s11val != -1 && !m_sf11.IsEmpty())
{
    qp = new CCeDBQueryProp();
    qp->SetType(CCeDBProp::Type_String);
    qp->SetIdent(1);
    qp->SetString((TCHAR *)(LPCTSTR)m_sf11);
    qp->setOperation(_ops[m_s11val]);
    q.addField(qp);
}
// NOTE: other query properties omitted, see listing
CCeFindDBDatabase theDB;
theDB.Open(TEXT("JCSExample"));
CCeFindDBDatabaseIterater *results = theDB.Find(q);

TRACE0("\nResults:\n");
for (CCeDBRecord *r = results->first(); r; r = results->next())
{
        CCeDBProp* strfield = r->GetPropFromIdent(1);
        CCeDBProp* intfield = r->GetPropFromIdent(2);
        CCeDBProp* datefield = r->GetPropFromIdent(3);
        FILETIME f = (datefield->GetFiletime());
        SYSTEMTIME t;
        FileTimeToSystemTime(&f, &t);
        TCHAR timeStr[256];
        GetDateFormat(LOCALE_SYSTEM_DEFAULT,DATE_LONGDATE,&t,0,timeStr,256);
        TRACE3("\t%s\t%d\t%s\n", strfield->GetString(),
            intfield->GetLong(), timeStr);
        delete r;
}
int hits, tests;
bool range;
results->stats(hits, tests, range);
TRACE3("Stats\n\tmatches: %d\n\tmisses: %d\n\tTotal: %d\n",
                                        hits, tests, hits+tests);
TRACE1("Range search %s used\n", range ? TEXT("was") : TEXT("was not"));
```

ACTIVE DATA OBJECTS

Listing One

```
<!-- USERLOGIN.HTM -->
<HTML>
<HEAD>
<TITLE>Application Login</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">

Please login with your user name and password.

<FORM NAME="LoginForm" METHOD="POST" ACTION="login.asp">
  <INPUT NAME="UserName" TYPE="TEXT" SIZE=25>
  <INPUT NAME="UserPwd" TYPE="TEXT" SIZE=25>
  <INPUT TYPE="RESET" VALUE="Clear">
  <INPUT TYPE="SUBMIT" VALUE="Log In">
</FORM>

</BODY>
</HTML>

<!-- LOGIN.ASP -->
<%@ LANGUAGE="VBSCRIPT"%>
<%
Response.Buffer = TRUE
'**********************************************************************
' This script handles the user login
'**********************************************************************

UserName = Request.Form("UserName")
UserPwd = Request.Form("UserPwd")
if UserName = "" or UserPwd = "" then
  Response.Redirect("userlogin.htm")
end if

cADOConnectString = "dsn=example_db;uid=user;pwd=userpwd"
set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open(cADOConnectString)

Query = "SELECT UserID from tbl_User where ((UserName = " & UserName _
  & ") and (UserPwd = " & UserPwd & "))"
set RS = Conn.Execute(Query)
if not RS.EOF then
  UserId = RS("UserID")
else
  Response.Redirect("userlogin.htm")
end if

RedirectURL = "welcome.asp?sesskey="
Query = "SELECT SessionID from tbl_Session where UserID = " _
  & CStr(UserId)
if not RS.EOF then
  RedirectURL = RedirectURL & RS("SessionID")
else
  DateTime = Date & " " & Time
  Query = "INSERT tbl_Session values(" & CStr(UserId) & ", '" _
    & DateTime & "'); SELECT @@IDENTITY"
  Conn.Execute(Query)
  set RS = RS.NextRecordSet
  RedirectURL = RedirectURL & RS(0)
end if

Response.Redirect(RedirectURL)
%>
```

Listing Two

```
<!-- WELCOME.ASP -->
```

(continued on page 112)

Every direction carries a solution

SAME COMPILERS FOR UNIX & WINDOWS NT

WINDOWS NT

- Secure path for future on Intel
- IA-32 versions today
- IA-64 at first system ship

FORTRAN 90

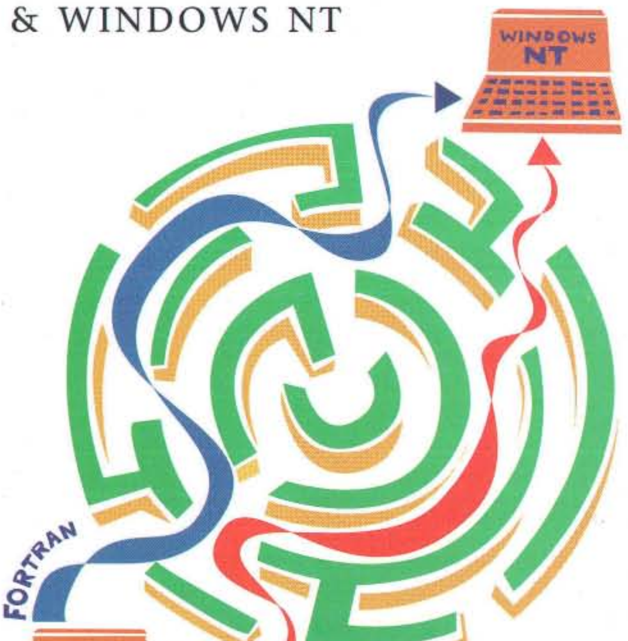
- F90 Standard Conformance
- HP, DEC, Cray & Microsoft Extensions
- Super-scalar & Parallel Optimizer
- Excellent run-time diagnostics

C++

- ANSI Draft Standard
- C++ targeted optimizations
- Standard + Rogue Wave Libraries
- Early dialects + ANSI C

UNIX

- Field proven for Intel, PowerPC, Sparc, Mips



Third party marks and brands are the property of their respective owners

EPC

CONTACT EPC FOR FURTHER INFORMATION:

EDINBURGH PORTABLE COMPILERS

USA	tel: (217) 398-5787	e-mail info@epc.com	http://www.epc.com
International	tel: +44 (0) 131 225 6262	e-mail info@epc.co.uk	http://www.epc.co.uk.

Putting You In Control...

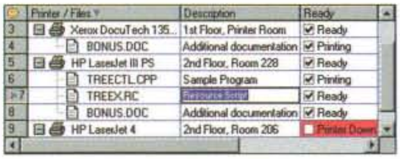
New

SftBox/DLL

Combo box with multiple columns, headers, column drag & drop, virtual data, and much more...

SftTree/DLL
SftTree/OCX

Single and multiple selection tree control, multiple columns, cell editing, headers and more grid-like features.



SftTabs/DLL

Add tabbed windows and tabbed dialogs to all your applications. Many tab styles and advanced features.

These royalty-free controls support Windows 3.1, NT and 95 in 16 and 32-bit applications.

softel vdm
NOT THE INDUSTRY STANDARD

(941) 505-8600
FAX (941) 505-8555

Call today for a free catalog!

318 Tamiami Trail, Suite 14 - Punta Gorda, FL 33950

www.softelvdm.com

PCYACC® Version 7.5

PROFESSIONAL LANGUAGE DEVELOPMENT TOOLKIT

Includes "Drop In" Language Engines for SQL, dBASE, POSTSCRIPT, HYPERTALK, SMALLTALK-80, C++, C, PASCAL, PROLOG, FORTRAN, COBOL, BASIC, SGML, ASN, RPG, REXX, PL1, SNA, RTF, VISUAL BASIC, SQL2, DB2, VHDL, HTML, VMRL, JAVA, ODMG-ODL/OQL, SQL3, MODULA-3, DELPHI, VBS, and ADA.

PCYACC Version 7.5 is a complete Language Development Environment that generates C, C++, Java, Delphi, Visual Basic, and VBS source code from input Language Description Grammars for building Assemblers, Compilers, Interpreters, Browsers, Page Description Languages, Language Translators, Syntax Directed Editors, Language Validators, Natural Language Processors, Expert System Shells, and Query languages.

- **Portable Object Oriented Classes for C++ and JAVA – Error, Symbol Table, Syntax Tree, Yacc, and Lex.**
- **Debugging tools include runtime Visual Parser Debugging, Abstract Syntax Tree generation, and Cross Referencing.**

CodeCheck® Version 7.5

SOURCE CODE ANALYST

Includes "Drop-In" Rules for Compliance analysis, Adherence to specifications, Measures of complexity, Silent error detection, Code maintainability, and Portability.

CodeCheck Version 7.5 is a programmable tool for managing all C and C++ source code on a file or project basis. CodeCheck is input compatible with all variants of K&R, ANSI C and C++. CodeCheck is designed to solve all of your Portability, Maintainability, Complexity, Reusability, Quality Assurance, Style Analysis, Library/Class Management, Code Review, Software Metric, Standards Adherence, and Corporate Compliance Problems.

- **Compliance – CodeCheck allows your corporate coding and project specification standards to be completely automated for compliance validation.**

CodeCheck includes pre-written expert system Rule Files that can be applied to any C or C++ project.

30 day Money back guarantee! Free AIR Shipping anywhere in the world!

www.abxsoft.com

	CodeCheck	PCYACC
DOS/WIN16	$495	$495
MAC	$495	$495
OS/2	$995	$995
WIN32 95/NT	$995	$995
UNIX/VMS	$1995	$1995

ABRAXAS™ Software, Inc.

5530 SW Kelly Ave., Portland, OR 97201 USA
TEL (503) 244-5253 - FAX (503) 244-8375
E Mail: sales@abxsoft.com

To Order Call **1-800-347-5214**

(continued from page 111)

```
<%@ LANGUAGE="VBSCRIPT"%>
<%
Response.Buffer = TRUE
'***********************************************************************
' This page welcomes the authenticated user
'***********************************************************************
if Request.QueryString("sesskey") = "" then
  Response.Redirect("userlogin.htm")
end if

cADOConnectString = "dsn=example_db;uid=user;pwd=userpwd"
set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open(cADOConnectString)

Query = "SELECT UserID from tbl_Session where SessionID = " _
  & Request.QueryString("sesskey")
set RS = Conn.Execute(Query)
if not RS.EOF then
  UserId = RS("UserID")
  Query = "SELECT UserFullName from tbl_User where UserID = " _
    & CStr(UserId)
  set RS = Conn.Execute(Query)
  if RS.EOF then
    Response.Write("Bad user ID in welcome.asp")
    Response.End
  else
    UserFullName = RS("UserFullName")
    DateTime = Date & " " & Time
    Query = "UPDATE tbl_Session set LastAccessTime = " & DateTime _
      & " where SessionID = " & Request.QueryString("sesskey")
    Conn.Execute(Query)
  end if
else
  Response.Redirect("userlogin.htm")
end if
%>
<HTML>
<HEAD>
<TITLE>Welcome Page</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">

Welcome <%=UserFullName%>.

</BODY>
</HTML>
```

HOTVIEWS

Listing One

```
Hv_VaInitialize(argc,           argv,
        Hv_NORAINBOW,           True,
        Hv_USEWELCOMEVIEW,      True,
        Hv_VERSIONNUMBER,       101,
        Hv_USEEXTENDEDCOLORS, True,
        NULL);
Init();
Hv_Go();
```

Listing Two

```
void Hv_GetAttributes(va_list ap,
              Hv_AttributeArray attributes,
              char **optlist,
              int *tags,
              int *fonts,
              int *colors,
              short *numopt)
{
  int                 argflag = -1;
  int                 prevargflag;
  /*give defaults for all attributes */
  Hv_SetDefaultAttributes(attributes);
/* now get the attributes */
  while (argflag != 0) {
    prevargflag = argflag;
/* the argflag is the keyword, #defined as a set of sequential ints such as
#define Hv_BACKGROUND 22, etc. the NULL terminator will produce a zero
argflag */
    argflag = va_arg(ap, int);  /* should be >= zero & < some max */
    if ((argflag < 0) || (argflag >= Hv_NUMATTRIBUTES)) {
    /* unintersting error handling omitted */
    }
    else if (argflag > 0) {
/* group according to type, place in the attribute array which is an array of
unions. We need insider information about what type is associated with each
attribute. */
      if ((argflag == Hv_DRAWCONTROL)
      || (argflag == Hv_USERDRAWCONTROL)
       /* or many integer attributes omitted */
      || (argflag == Hv_RELATIVEPLACEMENT))
         attributes[argflag].i = va_arg(ap, int);
      else if ((argflag == Hv_STATE)
      || (argflag == Hv_USEWELCOMEVIEW)
       /* or many short attributes omitted */
      || (argflag == Hv_PLACEMENTGAP))
          attributes[argflag].s = (short)va_arg(ap, int);
/* omitting code for other types: chars, strings, floats, etc. */
      else  /* only thing left are the pointers */
    attributes[argflag].v = va_arg(ap, void *);
    } /* end argflag > 0 */
  }
  va_end(ap); /* terminate the processing */
}
```

Listing Three

```
void Init() {
   Hv_canvasColor = Hv_lightSeaGreen; /*Change BG color */
   WindowTile();      /* tile the window with propaganda */
   MakeMenus();       /* modify the main menu */
   InitControls();    /* Initialize controls */
   AddLogo();         /* add a logo */
   InitQuickZoom(); /* initialize quick zooms */
```

AD LINK 124

```
    FinalInit();    /* final initialization */
}
```

Listing Four

```
     void MakeMenus() {
    Hv_Widget item;
    char *text;
/* create the menu items for the Help menu */
    text = (char *)Hv_Malloc(strlen(Hv_programName) + 20);
    strcpy(text, "About ");
    strcat(text, Hv_programName);
    strcat(text, "...");
    item = Hv_VaCreateMenuItem(Hv_helpMenu, Hv_LABEL, text,
                  Hv_CALLBACK, DoAboutDlog, NULL)
   Hv_Free(text);
/* add a "new view" item to the action menu */
    item = Hv_VaCreateMenuItem(Hv_actionMenu,
                   Hv_LABEL,      "New View",
                   Hv_CALLBACK,  GetNewView,
                   NULL);
/* add a plot canvas option to the action menu */
    item = Hv_VaCreateMenuItem(Hv_actionMenu, Hv_LABEL, "New Plot Canvas",
                   Hv_CALLBACK, NewPlotView, NULL);
     }
```

Listing Five

```
static void FinalInit() {
  Hv_View view;
  Hv_AddPath("./mapmaker");   /* add to search path */
  Hv_InitMaps("hv.maps");     /* initialize maps */
  view = NewView(MAPVIEW); }  /* create one initial view*/
}
```

Listing Six

```
Hv_VaCreateView(&View,
    Hv_TAG, tag,
    Hv_DRAWCONTROL,
    Hv_STANDARDDRAWCONTROL+Hv_SAVEBACKGROUND,
    Hv_POPUPCONTROL, popupcontrol,
    Hv_TITLE, tchr,
    Hv_INITIALIZE, ViewSetup,
    Hv_CUSTOMIZE, ViewCustomize,
    Hv_USERDRAW, ViewDraw,
    Hv_OFFSCREENUSERDRAW, OffScreenViewDraw,
    Hv_FEEDBACK, ViewFeedback,
    Hv_LEFT, left,
    Hv_TOP, top,
    Hv_XMIN, xmin,
    Hv_XMAX, xmax,
    Hv_YMIN, ymin,
    Hv_YMAX, ymax,
    Hv_HOTRECTWIDTH, width,
    Hv_HOTRECTHEIGHT, height,
    Hv_MINZOOMWIDTH, minw,
    Hv_MAXZOOMWIDTH, maxw,
    Hv_MINZOOMHEIGHT, minh,
    Hv_MAXZOOMHEIGHT, maxh,
    Hv_SIMPROC, (Hv_FunctionPtr)SimulationCB,
    Hv_SIMINTERVAL, 2000,
    Hv_HOTRECTVMARGIN, 7,
    Hv_HOTRECTCOLOR, Hv_blue-2,
    Hv_MOTIONLESSFB, True,
    Hv_MOTIONLESSFBINTERVAL, 500,
    NULL);
```

Listing Seven

```
Choice = Hv_VaCreateItem(View,
    Hv_TYPE, Hv_CHOICESETITEM,
    Hv_RELATIVEPLACEMENT, Hv_POSITIONBELOW,
    Hv_PLACEMENTITEM, Box2,
    Hv_PLACEMENTGAP, 1,
    Hv_NOWON, mdata->projection,
    Hv_FILLCOLOR, Hv_powderBlue,
    Hv_ARMCOLOR, Hv_navyBlue,
    Hv_COLOR, Hv_gray10,
    Hv_FONT, Hv_fixed2,
    Hv_OPTION, "Mercator",
    Hv_OPTION, "Orthographic",
    Hv_SINGLECLICK, ProjectionChoiceCB,
    Hv_TITLE, "Map Projection",
    NULL);
```

Listing Eight

```
Item = Hv_VaCreateItem(View,
    Hv_TYPE, Hv_WORLDPOLYGONITEM,
    Hv_TAG, DCITEM,
    Hv_NUMPOINTS, np,
    Hv_DATA, fpts,
    Hv_NUMROWS, 1,
    Hv_NUMCOLUMNS, 1,
    Hv_DOMAIN, Hv_INSIDEHOTRECT,
    Hv_DOUBLECLICK, EditDCItem,
    Hv_DRAWCONTROL, Hv_ZOOMABLE + Hv_INBACKGROUND,
    Hv_AFTEROFFSET, DCAfterOffset,
    Hv_USER1, (int)superlayer,
    Hv_USER2, (int)sect,
    Hv_BALLOON, (void *)text,
    Hv_FIXREGION, FixDCSuperLayerRegion,
    NULL);
Item->type = Hv_USERITEM;        /* redefine as user item */
Item->standarddraw = DrawDCItem; /* redirect std. drawing */
```

DDJ

"This method of delivering training is a great way of getting our employees hands-on education in new technologies. Implementation was quick: our workers could sign up on Monday and be going through the coursework on Tuesday."

- *Brad Clayton*
Training Manager, Cirent Semiconductor

With DigitalThink training, you don't suffer travel or meeting room costs. You never get outdated training. And you don't struggle with support issues – unlike CD-ROMs, there are no special software installations or hardware requirements. All a student needs is a Web browser and an Internet or intranet connection.

For a course evaluation and more information on our corporate programs, contact our Sales Department at sales@digitalthink.com or call 415.437.2800.

DigitalThink
LEARN WHAT YOU WANT

www.digitalthink.com

©1998 DigitalThink. All rights reserved.

AD LINK 413

Build on a Rock Solid Test Foundation

LabWindows™/CVI

Virtual Instrumentation Tools for C/C++

When you need to build a BIG test system...

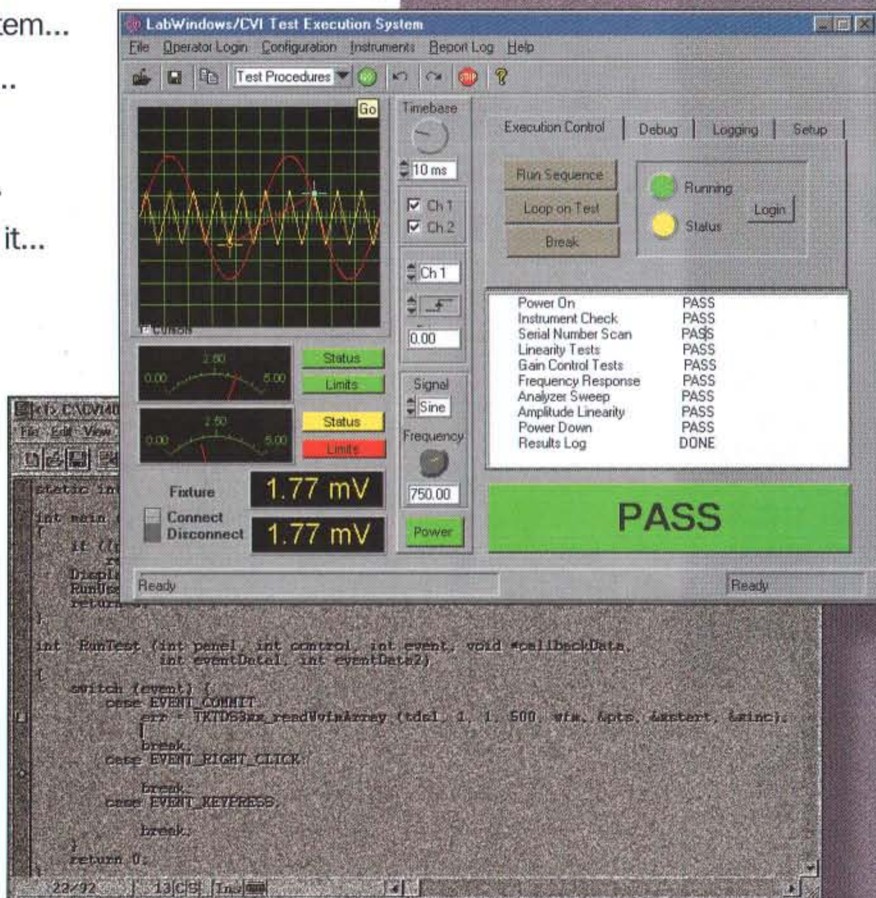
When you can't afford to get locked in...

When it has to be fast...

When you need a solution that works...

When your production line depends on it...

When your job is on the line...



Why look anywhere else?

- ✔ Standard ANSI C Programming
- ✔ Lightning-fast code generation
- ✔ Open development environment
- ✔ VXI*plug&play* driver standard

For data acquisition, GPIB, and VXI instrument control, analysis, and user interface tools, LabWindows/CVI has everything you need under one roof.

Put LabWindows/CVI to Your Test!

AD LINK 138

NATIONAL INSTRUMENTS™

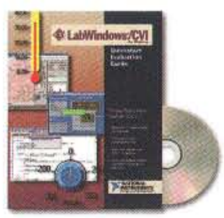
The Software is the Instrument™

U.S. Corporate Headquarters

Tel: (512) 794-0100 • Fax: (512) 794-8411

info@natinst.com • www.natinst.com

Worldwide network of direct offices and distributors.



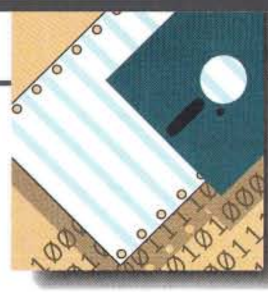
Call today for your FREE LabWindows/CVI Evaluation Version

(800) 661-6063

(U.S. and Canada)

© Copyright 1997 National Instruments Corporation. All rights reserved.

Product and company names listed are trademarks or trade names of their respective companies.



A Chat with Bob Bemer

Michael Swaine

As the millennium looms, a computer pioneer has come out of retirement to create a product he claims will solve the Year 2000 problem. And he claims it will do so an order of magnitude faster than other approaches.

This veteran programmer knows all about the Year 2000 problem, because he helped create the Cobol language in which much of the problem code was written, he created many of the standards for data encoding that have existed since the early days of computing, and he wrote the *Datamation* article back in the 1970s that alerted many to the two-digit date problem that others are just waking up to now.

Bob Bemer is a programmer from the generation of legends: He worked with Fortran creator John Backus and COBOL creator Grace Murray Hopper and IBM 360 designer Fred Brooks. He was enjoying a well-earned retirement when he hit upon his idea for solving the Year 2000 problem. While another senior citizen might have been satisfied to pass the idea on to a younger programmer, Bob moved to Texas, set up a company, developed a software product, and started talking to the press. I recently talked with Bob about his past contributions to computer programming and his proposed solution to the Y2K problem. Here's what he had to say.

Don't ASCII, Don't Telly

***DDJ*:** When I think about the standards you are responsible for and the projects you have worked on in your career, it seems that the name "Bob Bemer" ought to be a household word among programmers. But I suspect that's not the case. Maybe after this interview, that will change a little.

You're known, among those who have heard of you, as the inventor of ASCII, the inventor of the escape sequence, the guy who named Cobol, a pioneer in word processing and time sharing and international standards for data processing. I want to hear about your Year 2000 solution, but I'd also like to know how you came to invent ASCII.

Michael is DDJ's *editor-at-large and can be contacted at mswaine@swaine.com.*

BB: I made a survey in 1960 [while working for IBM] and found out there were over 60 different ways the alphabet was coded for various computers. So I started to pick out the problem of interchanging files, and I started making proposals for a single code. Before that I did the character set for the Stretch machine at Los Alamos. That was the first eight-bit-byte computer that I know of. But I made a mistake. I put the alphabet in as capital A, lowercase a, capital B, lowercase b. And that was stupid. It was then that I wound up with the escape sequence idea that I published sometime in 1960.

***DDJ*:** But ASCII became more than an IBM standard. How did the internationalization of ASCII come about?

BB: I was invited to talk to the British Standards Institution and I got to go to the electronic industry association. And finally I was called by two IBM vice presidents. They said they would like to revitalize what was then called the OEMI— the Office Equipment Manufacturer's Institute— and they wanted proposals from me for what should be done in the way of computer standards. We had a big meeting, the first meeting of X3 under ANSI auspices. And that later became an ISO committee when we went international with the standards for computers.

Then came the fateful day. We had ASCII going. We were about to sign off and the 360 was about to get out. And Freddie Brooks tells me that the printers and punches are not ready in ASCII.

***DDJ*:** They were still designed for EBCDIC?

BB: Right. And one manager, now dead (but I won't say "God rest his soul") decided they were going to do both. They would put in p bit, and if the p bit was 0, the machine would run in EBCDIC. If the p bit was 1, it would run in ASCII. He thought that was a reasonable way to solve the problem, because he had to announce the 360 in a hurry. So they did. Unfortunately nobody told the programmers, and they did all their systems programming in EBCDIC. As a result, they couldn't make the thing run in ASCII. So ASCII originated at IBM, but they didn't follow through with it. Isn't that a crazy story?

Grace Under Pressure

***DDJ*:** Okay, I also have to ask, what was your involvement with Cobol?

BB: I came to IBM in late '55 to [write] a system that allowed you to use the 705 commercial computers for scientific work. It was a big success, but when I did that, I was in the same room as John Backus. And I was watching the Fortran work, and saying, hell, this has got to happen.

So I got hold of [A.J.] Perlis and he allowed us to use his compiler from Carnegie Tech. And we chucked that in under a thing we called "Fortransit," which is the first time we had a programming language that worked on different computers.

And in January of 57, I got to meet Dr. Grace Hopper of the Franklin Institute in Philadelphia. And I started working on commercial translators. And we sort of blended all those things together and came up with Cobol.

Probably it was a good thing in one way, but it was a mistake in other ways because not everybody who got in the act of programming computers was careful to annotate it. People made mistakes. Like in computing the year.

***DDJ*:** What was your main contribution to Cobol?

BB: I guess the major thing I did for Cobol was the picture clause.

***DDJ*:** Which is?

BB: The picture clause? That's where you say, I have a piece of data, this is its domain, and these are its characteristics. If you said S99, that was going to be a two-digit number, signed. And we used various other symbols to say this is all alphabetic, punctuation, and the like. This is the first data typing. If you look at some of the later languages like ADA, PL1, everybody brags about heavy typing, you know, very strict. Control programmers against their own mistakes. Well, that was the first data typing.

Need a Modern Parsing Tool??

Visual Parse++ is the only tool that brings cutting edge parsing technology to all developer's. You can design any parser without ever leaving our sophisticated visual development environment. Whether you're new to parsing technology, or an experienced veteran, Visual Parse++ is the right tool for you.

Visual Parse++ provides native programming support for **C, C++, Java, Visual Basic, and Delphi.** Our C++ and Java classes are fully supported on all platforms. Visual Parse++ also comes with drop-in parsers for **HTML, SQL, RTF, Java, C, C++,** and more.

New to version 3.0 is **natural language parsing**. Visual Parse++ is easily the most powerful parsing engine in the world with generalized parsing capability. Visual Parse++ now can handle any grammar, including ambiguous grammars. This opens up the area of **natural language parsing**, among other applications. Also included is a brand new user interface, with many new features and improvements, along with 3D parse tree and machine views.

Java | C++ | Delphi | ActiveX | VB

Order a copy today!

Call 800-988-9023

www.sand-stone.com

SandStone Technology

Software Development Productivity Award 1995

939 Coast Blvd
Suite 4C
La Jolla, CA 92037
Phone: (619) 454-9404
Fax: (619) 454-9467
willd@sand-stone.com
www.sand-stone.com

Bob Bemer

***DDJ*:** So you invented data typing?
BB: I never thought of that. I didn't [actually] create the picture clause, I [really] created data typing.

Don't Blame the Programmers

***DDJ*:** All right, let's talk about the Year 2000 problem. I've heard all sorts of estimates for the magnitude of the problem. What's yours?
BB: I talked to a Gartner Group guy last November, and asked, how's your $600 billion dollar estimate holding up? And he said, real fine. When people start talking about it getting into the trillions, I won't be surprised, except I'm going to try to bring it down. If the U.S. got our act together 100 percent, we'd still be in terrible shape if the Asians didn't and the Europeans didn't. And the Europeans are trying to do monetary conversion to the Euro and the Year 2000 [solution] at the same time.

I heard from a guy at Microsoft that they moved the clock ahead on the equipment up at the Hannaford atomic energy place and they blew all the air conditioning, they've gotta get new air conditioning.

***DDJ*:** That's scary.
BB: We know there's a lot of nuclear stuff that's gone haywire. A couple of months back, I talked to the gal who's running the Year 2000 program for the State of Texas, and they have a nuclear reactor, but she admitted she doesn't know what they're doing.

***DDJ*:** The naïve question I sometimes hear from people is, how hard can it be to find references to dates in source code and change them? I guess one of the several ways that question is naïve is that it assumes the source program is even available.
BB: Source code is not around in about 30 percent of cases.

***DDJ*:** And in some cases people may have the source but it's not the latest version. Or maybe the code hasn't been recompiled since the last time the compiler was revved and might break just as a result of recompiling.
BB: Yup. There are a lot of mismatches out there.

***DDJ*:** And, of course, it's both the programs and the data.
BB: And when you try to stretch it out with expansion, you know, put in an extra 19 and an extra 20, everything gets shoved over. And those poor compiled programs, particularly the object programs, don't know where anything is anymore.

***DDJ*:** So we are in this mess all because a bunch of Cobol programmers wanted to save a few bytes of memory?
BB: No. Everybody blames the programmers, but it wasn't their fault. It was people [like managers, customers]. They were happy they didn't have to use 19s. There were international standards, [but people wanted to use] month-day-year instead of year-month-day, like computers had to have it.

This complicated the problem tremendously. If you write the year first, you don't have to go through a lot of fancy computations. People say that they didn't have enough memory, that it was expensive. Well, if they'd used the proper order they would have had plenty of memory because they could have saved all the routines they had to have to jockey it around for compares and to write it on this or that report. I get pretty vehement about that. As a programmer, I don't think we're at fault at all.

As in Bigit Baudot?

***DDJ*:** You have a solution to the Year 2000 problem, but your solution is different from anything else out there. Apparently you don't think much of approaches like date-field expansion and windowing.
BB: Date-field expansion will never make it. They thought so a year and a half ago, but now they don't have time.

It means you actually rewrite the Cobol program and say pic999 instead of pic99 and if you're lucky you'll find all the things in the program that are dates. But as a practical matter you can't. I use the example of an insurance policy number where they've used the date of issue as part of the policy number. You can't insert a 19 in an insurance policy number.

***DDJ*:** And windowing?
BB: Also flawed. For one thing it's evanescent. Another thing is a lot of people want sliding windows and by that time you don't know what's going on.

And then you've got these 28 guys. You know that one? You know what happens? Well on IBM equipment when you say pic99, boy, that's always gonna be a positive number, right? Well, take my birthdate, 1920. That's stored as 20, you subtract 28 and you get minus eight and the compiler says, I know this value is always positive but just to be safe, I'm gonna overwrite it to make it positive, and I have every right

to do that. And now I'm a plus eight, right? And on the way out, you add 28 back, and I was born in 1936. Well, I can use the extra years, but it's just not gonna work. And there are people out there selling that.

***DDJ*:** What you've developed is an object-code solution. You piggyback millenium and century marks onto the two year bytes. As I understand it, you tweak the data in this way and then you sidetrack the object code that deals with dates into subroutines that understand this tweaked data. But maybe you'd better explain it.

BB: We've made a microprogram that will operate on Bigit arithmetic. Bigit is short for "Bemer digit."

***DDJ*:** Bigitizing data is what I called tweaking it. How does that work?

BB: We put a millenium century indicator over the decade digit so we can handle dates from 1600 up to 2099. The problem, of course, is that the hardware won't handle it. So whenever we think we're going to come upon one of these things we say, hey we have to handle that by a subroutine. And we examine the object program. First we examine the opcode and if it says drive the printer, we say, hey we're not doing any arithmetic on years. If it says add and it's a decimal add, we derail that. But only if we look at the operands and they're of appropriate length. If it's an add of two 16-digit operands, we say that has nothing to do with years and we don't derail it. But every time we come across an opcode that could possibly be an [appropriate] arithmetic instruction, we derail it to this subroutine and we put it itself in a table of derailed instructions.

So when the object code runs, it hits this derail, indexes over the instructions in there, plays like a regular accumulator in a machine, goes out and gets the operands, inspects them, and if one of them has a Bigit and you're making a comparison and the other doesn't have a Bigit in it, it says, wait a minute. How can you be comparing a year value to a nonyear value? Maybe that should be a Bigit, too.

When we get down to that point we expand it to either 16, 17, 18, 19, or 20, do the arithmetic, recompress it, put it back where the answer is supposed to go and then come back to the original machine. I describe this as following the Napoleonic code where you're guilty until proven innocent.

But when you are proven innocent, when we find that it's not [date arithmetic], we take it out of the loop and put the original instruction back where it was. And thus we've reduced any time we can. However we've got a lot of people who say we don't care if it does add time; just fix it. But, as a matter of fact, we don't add all that much time. Can't detect it.

***DDJ*:** And doesn't the Bigitized data look the same as the old data to old unmodified programs?

BB: That's another nice thing about this, as it applies to IBM equipment at least. They use packed decimal instructions, and when you pack a number only [one] zone gets [retained]. All the other zones disappear. If you run the old program against the old data, that's what you had, right? If you run the enabled program against the old data, there isn't any Bigit data there, so the program runs exactly as before. Now here's the kicker: When you run the old program against the new data, it doesn't see the decade zone because it gets tossed away in the packed decimal operation. And you still get the same wrong answer. Or if you were getting correct answers, no problem.

What that means is that you can put up one application at a time. All the other approaches make you change all your applications and databases simultaneously.

So, even if the first application Bigitizes the data, the other applications can still run as they were. They may be wrong but at least they're running.

***DDJ*:** So where does the order of magnitude advantage over other approaches come from?

BB: We don't touch the source program and we don't cause any errors to be made in the source program. So we don't have this 50 percent testing time that everybody else has. We can test statistically. If it ran before there's good odds it'll run again if our device works properly. And instead of m users times n programs, we only have to check one program— ours.

Now if your testing time goes down from 50 percent to 5 percent, you've got time before 2000 to go through [the code] in case there's any really weird stuff from programmer stupidity or clever tricks.

***DDJ*:** Thanks Bob.

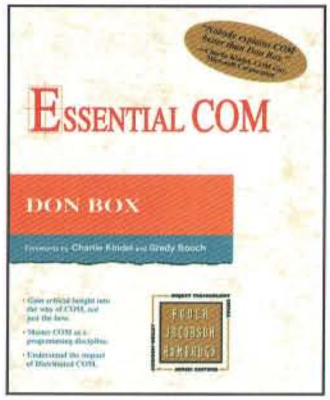
Bob's approach won't help you if you're working on PCs or with embedded systems: It's strictly a mainframe fix, at least so far. But that's where Bob's expertise is. There were a lot of things Bob and I didn't have time to get into, such as how his software figures out what data needs to be Bigitized. You can read Bob's many articles on the Year 2000 problem and his product, Vertex 2000, at the BMR Software web site (http://www.bmrsoftware.com/).

By the way, it's now a little easier to remember the URL for my web site, where I keep updated links, corrections, and reader feedback for this column. It's http://www.swaine.com/. And my new e-mail address is mswaine@swaine.com.

DDJ

It's a jungle out there!

Read *Essential COM* to ensure your survival.



0-201-63446-5, 464 pages, $34.95

Don Box, cofounder of DevelopMentor and well-known for training events like Guerrilla COM, knows COM can be challenging. That's why he wrote *Essential COM*. Don's book explains the why of COM not just the how. Master COM by learning to apply the model creatively and effectively – pick up *Essential COM* today!

"Nobody knows COM better than Don Box."
–Charlie Kindel, COM Guy, Microsoft Corporation

"Don Box makes it possible for mere mortals to join the COM cognoscenti. If you're a C++ COM programmer, buy this book."
–David Chappell, Principal, Chappell & Associates and author of Understanding ActiveX and OLE

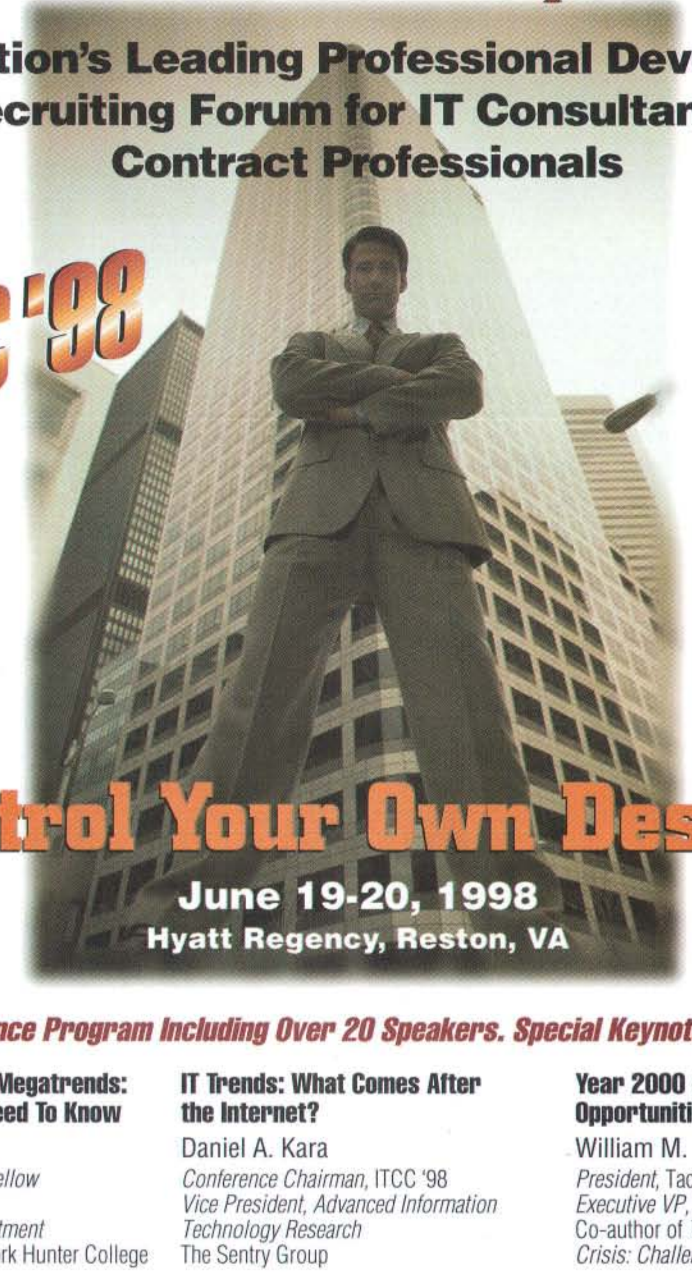
Addison-Wesley

Check out the table of contents and sample chapters at:
http:www.awl.com/cseng/specials/ddj498.html/

Find this book wherever technical books are sold or by calling (800) 822-6339.

AD LINK 297

IT Consultants & Contractors World Conference & Exposition



Join over 50 of the Region's Top IT Staffing and Software Companies at the ITCC '98 Expo, along with a Two Day Conference Program and Workshops for maximizing your consulting success…

Two Day Conference Program Including Over 20 Speakers. Special Keynote Speakers include:

IT Staffing Industry Megatrends: What Consultants Need To Know

Dr. Howard Rubin
META Group Research Fellow
Chairman of the
Computer Science Department
City University of New York Hunter College

IT Trends: What Comes After the Internet?

Daniel A. Kara
Conference Chairman, ITCC '98
Vice President, Advanced Information Technology Research
The Sentry Group

Year 2000 & Beyond: Opportunities for IT Consultants

William M. Ulrich
President, Tactical Strategy Group, Inc.
Executive VP, Triaxsys Research LLC
Co-author of *The Year 2000 Software Crisis: Challenge of the Century*

For a Free Brochure Call: 508-870-5858 or Visit our Web Site: www.itccexpo.com

Sponsor
TRI Group, Inc.

Corporate Co-sponsor
AJILON
INFORMATION TECHNOLOGY SERVICES

Exclusive Association Co-sponsor
NACCB
National Association for Computer Consultant Businesses

User Group Co-sponsors
DC Area Sybase Users Group
Powersoft Users Group

Premiere Media Co-sponsor
CONTRACT PROFESSIONAL

Media Co-sponsors
COMPUTERWORLD career@gent
Dr. Dobb's Journal
THE SENTRY GROUP
Contract Employment WEEKLY
DBMS
SOFTWARE MAGAZINE
Computer Consultants & Contractors Newsletter

MIDIFile: Standard MIDI Format File Parsing

Al Stevens

Recently, I undertook some projects that involved reading and processing data from files written in the Standard MIDI Format (SMF)—an architecture that, among other things, records MIDI messages in data streams, also called "sequences." Sequences of MIDI messages are what a sequencer program or device sends in real time to electronic musical instruments to play a production. SMF prescribes a standard format to record the real-time messages on disk, as well as other information—tempo, for example—that a playback system needs to accurately reproduce the production, and information meaningful to the human MIDI programmer—key signature, time signature, and so on. So, irrespective of a particular sequencer's internal representation of these data, if the sequencer can generate an SMF file, other sequencers can play back the production.

Furthermore, a computer program that can read an SMF file can process MIDI production data in support of various applications, which is the point of this discussion.

Each message in an SMF file represents an event that a MIDI system processes in the course of the playback of a musical production. There are three kinds of events: MIDI events, metaevents, and system-exclusive events. Each event includes a time stamp that specifies when the event is to be processed. It is expressed as a delta number of clock ticks counted from the beginning of the track in which the event is stored. More about tracks later.

MIDI events are real-time messages. They play notes on multiple simulated instruments, specify "aftertouch" values, encode controller events (volume, pan, and sustain pedal, for example), process pitch bends, assign channels to instrument sounds (patches), control lighting devices, and so on.

Al is a DDJ *contributing editor. He can be contacted at astevens@ddj.com.*

Metaevents specify such things as tempo, how to relate clock ticks to tempo, time signature, key signature, channel assignments, and text fields to name the song, the instruments, and the tracks.

System-exclusive events contain data defined by the manufacturer of the device or program that processes the SMF file.

An SMF file comprises a header chunk followed by one or more track chunks. There are three SMF formats.

- Format 0 is a single-track file with all the channel information and metaevents mixed in the single track.
- Format 1, the most common format, contains a track of metaevents and one or more tracks that contain channel information. Typically, each channel (instrument) has its own track, but nothing says that it must. Format 1 SMF file tracks are processed together. The delta time fields of each are relative to the beginning of the production.
- Format 2 is a multiple-track file wherein the tracks operate independently of one another—each track can represent a different production.

All of this suggests that the SMF architecture is complex, bewildering, and begging to be encapsulated so programmers can ignore the details and work at higher levels of abstraction.

MIDI tends to embrace an open, cooperative culture. Much of the software that MIDI users need is available as shareware or freeware. An Internet search revealed two freeware C-function libraries that parse MIDI files. One library is called Midifile (http://www.nosuch.com/midifile/midifile.zip). It is authored by Tim Thompson and augmented by Michael Czeiszperger. The other library is also called Midifile (http://www.ooblick.com/software/midifile.tar.gz) and is a rewrite of Thompson's library by Andrew Arensburger. He rewrote the library because he "wasn't quite satisfied with [Thompson's] implementation." Both libraries were written in the 1980s in K&R C, and both work in essentially the same way, with these differences: Arensburger's rewrite does not include Czeiszperger's enhancements to write SMF files; the rewrite does not support system-exclusive events; the rewrite sends error conditions to the library user whereas the original program simply exits the program upon encountering the first error. (Tim's site at http://www.nosuch.com/ includes lots of other MIDI software that programmers might find interesting. In particular, check out his KeyKit MIDI programming environment.)

The Midifile API of both libraries for parsing SMF files comprises a set of function pointers. An application assigns the addresses of custom functions to the pointers and calls a library function to begin parsing the file. The library processes the file and calls each function as the parser encounters the header, the tracks, and the events in the file. It is up to the using program to do something meaningful with the data. If the application does not initialize a pointer, the event associated with that pointer is parsed and bypassed. One notable feature of the Midifile API is the requirement for the application to manage opening, closing, and reading bytes from the SMF file. One of the function pointers expects to point to a user-provided function that returns the next sequential byte in the file.

Czeiszperger's enhancements permit an application to write SMF files. You initialize pointers to write characters to the stream and to write tracks. Then you call a function that writes the header and calls your track function for each track. From within that function, you call functions to write events.

Midifile contains all of the functionality that I needed for my project and more, so I decided to use the two libraries as inspiration for a rewrite in C++.

The *MIDIFile* Class

Besides representing a complex problem domain in need of object-oriented encapsulation, there are two facets of both Midifile implementations to which C++ is particularly well suited. First, error processing is better handled by C++ exception handling than by the common C idiom of returning error conditions. Second, the function pointer API mechanism is better represented by virtual functions in a C++ base class than by a set of global function pointers that the library user initializes. With C++, the application derives a class from an abstract base class that encapsulates the details of parsing the SMF file. The base class contains virtual member functions that the file parsing algorithm calls for each of the events. The derived class overrides only those virtual functions related to the events that the application wants to view. For example, a MIDI jukebox application would not need to see the time and key signature metaevents, but would need to see the tempo metaevent. The same application might want to display the text that identifies the production—the song title, for example, but would not be interested in the instrument name text that identifies each track.

The complete project—including all source code and a MIDI song file that one of the example programs creates—is available electronically (see "Resource Center," page 3). Midifile.h, for instance, is the header file that defines the *MIDIFile* class. *MIDIFile* is an abstract base class. To use it for parsing or generating an SMF file, an application derives a class from it, provides some arguments, and overrides some virtual functions. The *MIDIFile* member functions are defined in a file named midifile.cpp, which is not published here because of its length.

The file begins with *typedefs* that define the aliases, *Short* and *Long*. The purpose for these *typedefs* is to ensure that certain data types are 16 and 32 bits in width, necessary data types for some of the internal SMF data representations. When you instantiate an object of a class derived from *MIDIFile*, the constructor makes the following assertion:

```
assert(sizeof(Short)==2&&sizeof(Long)==4);
```

The next part of midifile.h declares the exception classes that *MIDIFile* throws. First is a *#define* macro named MFX that abbreviates the declarations. Following that are invocations of MFX to declare classes derived from *std::runtime_error* and to provide text to describe the nature of the exceptions. (Whenever people deprecate the C++ preprocessor and wish it would go away, I think of handy constructions such as the MFX macro, and I smile.)

A list of *const* declarations define symbols for the MIDI events, and an *EventBuffer* class manages the allocation of memory to hold the data values of variable-length events.

Next is the *MIDIFile* class declaration. Objects of the class can be instantiated to parse an existing SMF file or to create a new one depending on which constructor you use. Each constructor initializes a pointer to either an *ifstream* or an *ofstream* object and initialized the other pointer to zero. Subsequent member functions use these values to read or write the SMF file and also to determine whether the object was instantiated for input or output. The constructors and destructor are protected to ensure that you derive a class from *MIDIFile* rather than instantiating an object of *MIDIFile*. (Well, almost; a class derived from *MIDIFile* could itself instantiate a *MIDIFile* object, but there would be no reason to do so.)

MIDI events are real-time messages

MIDIFile includes private member functions to manage the way that SMF files store numerical values. SMF uses a byte order that is the reverse of that used by Intel processors, so some juggling is necessary when reading and writing those formats between memory and the file. Some values are stored in two bytes. The song's tempo is stored in three bytes. The header length and track length fields are stored in four bytes. All values are treated as twos-complement signed integers. Event data lengths and delta times are stored in a variable-length format wherein all but the last byte has the most significant bit set to one. Only bits 0–7 of the bytes in a variable-length integer contain numerical data. If you download the project, you can view these conversion functions in midifile.cpp.

Reading an SMF File

An application reads an SMF file by first deriving a class from *MIDIFile*, overriding virtual functions that process each of the *MIDIFile* components. The constructor of the derived class passes to the base *MIDIFile* class constructor a reference to an open SMF file. The *MIDIFile* constructor has a second argument, a *bool* that tells *MIDIFile* whether to bypass system-exclusive events (*true*) or to allocate memory for them on the heap and process them (*false*). System-exclusive events can involve large memory allocations. This mechanism permits a program to ignore them.

The application opens the SMF file and instantiates an object of the derived class. The application then calls the *MIDIFile::ReadMIDIFile()* function, which parses the file, calls the virtual functions for each event, and throws exceptions when it finds errors in the file.

What the application does depends on which virtual functions it overrides and what it does with the SMF data. Miditest.cpp (available electronically) is an application that overloads all the virtual functions and displays the contents of the SMF file on the console. The program catches exceptions and displays their text on the console. As such, miditest is a good utility program for testing the validity of SMF files and reporting what and where the problems are. Actually, if I had not written this program first, getting the next one to work and debugging the part of Midifile that creates an SMF file would have been much more difficult.

Each overridden function represents one part of the SMF file. The first function called is the *Header* function. Next is the *StartTrack* function for the first track. After that come the metaevent and MIDI-event functions for the track with the *EndOfTrack* function signifying that there are no more events in the track. The track and event functions repeat for each track in the file.

Each overridden function is passed a delta time value as its first argument. If the application overrides all the functions, the delta value is the same one recorded in the file with the events. Otherwise, the value passed is the delta time since the last event that was intercepted by an overridden function. An overridden function must not call the base class function it overrides. If it does, the delta time computation is compromised.

Writing an SMF File

An application writes an SMF file also by deriving a class from *MIDIFile*. In this case the constructor passes to the *MIDIFile* constructor a reference to an *ofstream* object and variables that specify the SMF format (0, 1, or 2), the number of tracks, and a value that specifies how many clock ticks there are in a quarter note. The derived class overrides the *StartTrack* virtual function. The overriding function uses the track number to determine which track is to be written, and calls *MIDIFile* member functions to write each track's events.

Listing One (listing begins on page 137) is playmidi.cpp, an application that

generates a simple SMF file that plays the first part of the "William Tell Overture" (no royalties to pay, that's why) on the piano and drums. The SMF file is included in the files you can download with this project. The application derives the *WmTellOverture* class from *MIDIFile* as explained previously, opens the *ofstream* file, instantiates an object of the class for the file, and calls the *WriteMIDIFile* function. Observe that the *MIDIFile* functions that *WmTellOverture::StartTrack* calls to write events to the track are the same virtual functions that an application overrides to read an existing SMF file.

MIDIfile Enhancements

Even though the *MIDIFile* class encapsulates much of the SMF file processing, you still view it from a relatively low level. The application has to know how to convert between musical and SMF notation. For example, an application has to know that the denominator of a time signature is really a power of two so that a march in 6/8 time is expressed as 6,3 (six and two raised to the third power). I don't know why, but that's how they did it. Key signature is a number between –7 and +7 to specify the number of flats or sharps and a boolean integer to indicate whether the key is minor (1) or major (0). Tempo and event time is another enigma. The header specifies the number of delta field clock ticks in a quarter note unless the number is negative, in which case it represents the number of ticks in a second. The tempo field itself is the number of microseconds in a quarter note. Notes are just numbers rather than in notation that musicians would understand. There's more, but I won't go into it here. The point is that to use Midifile, you have to understand these algorithms. A class library at a higher level of abstraction would understand musical notation, implement that in the API, and do the conversions for you. As I use this library, I expect to understand better where further encapsulation will make it more usable.

The C++ Programming Language, Third Edition

Bjarne Stroustrup's *The C++ Programming Language*, Third Edition (Addison-Wesley, 1997) has been available for several months. This work, by the creator of C++, is the definitive treatment of the subject and has been since its first edition in 1987. I must confess that I did not care for the first edition. I had expected a tutorial approach as elegant as the classic K&R white book. But then, K&R was about C, a programming language that supported a familiar programming model. The C++ programming model was new to most of us ten years ago, and Stroustrup's first edition was daunting, to say the least. Looking at it now, I find it far less so and much easier to read.

Comparing the first and third editions of *The C++ Programming Language* provides insight into how the C++ language has grown and changed in the past decade. The third edition has almost three times the number of pages and a slightly different organization. Whereas the first edition included a 67-page language reference manual at the end, the third edition includes only a language grammar section to represent formal language definition. This is appropriate. The ANSI/ISO Standard document, which is now the formal language and library definition, is itself about 750 pages long. Stroustrup plans to publish *The Annotated C++ Language Standard* (coauthored by Andrew Koenig, the ANSI C++ committee's Project Editor) sometime this year.

> MIDIFile *includes private member functions to manage the way that SMF files store numerical values*

The third edition takes a tutorial approach with many of Stroustrup's personal programming philosophies. The author's explanations of how he uses language features provide examples for learning the behavior of those features. He also explains code idioms that some programmers routinely use but that he finds inappropriate.

As much as possible, the third edition reflects Standard C++. When small language features are found to be missing, particularly new ones, Stroustrup pledges to add them to a future printing.

The book includes many code examples. There is no diskette or CD-ROM, because Stroustrup avoids a teaching approach wherein readers compile and run examples. His examples are mostly code fragments that demonstrate the points he makes and the issues he addresses. The code fragments are readable, meaningful, and neither frivolous nor cute, and since you do not compile them, you need not worry that your compiler does not fully support Standard C++.

There are four parts to the body of the book: "Part I: Basic Facilities;" "Part II: Abstract Mechanisms;" "Part III: The Standard Library;" and "Part IV: Design Using C++." Even if you are already a seasoned C++ programmer, Part IV, which is a rewrite of several chapters from the second edition, is worth the price of the book. It describes Stroustrup's philosophies on the design and development cycle of a software project involving C++. In his words, Part IV aims "to bridge the gap between would-be language-independent design and programming that is myopically focused on details."

The three appendixes are: "A: The C++ Grammar;" "B: Compatibility;" and "C: Technicalities." Appendix B discusses the differences between C and C++ and explains how the languages have become more compatible over time. Some of this convergence results from changes being made to the C specification (double-slash comments and no implicit *int*, for example). The appendix also discusses the issues related to porting C++ code from older C++ implementations, advising that, where possible, you should use the latest implementation of a compiler so that newer features are available to you.

Appendix C is about technical details that a programmer faces that are not necessarily language issues. I particularly like the discussion on the problems associated with traditional multidimensioned arrays as compared to using STL containers to achieve the same result without the headaches.

This book is an essential addition to a C++ programmer's library. It is not for dummies, and it wouldn't be my first choice for an entry-level, self-help tutorial on C++ for beginning programmers. It is, however, an excellent textbook for programmers who are self-motivated and students who study under the watchful care of a skilled instructor. As an experienced C++ programmer, I find the book useful as a reference to language usage and behavior. The author invented the language and then stayed close to the standardization and innovation process for the duration, always maintaining a careful vigilance over the evolution of his brainchild. Consequently, this book serves, for those who do not care to pore over the ANSI/ISO document (or the promised annotated version), as the authority on the Standard C++ language, how it works, and how you should use it.

DDJ

(Listing begins on page 137.)

How Do I Create a Streaming Audio Java Applet?

L. Richard Moore

Streaming audio refers to audio that can be downloaded at the same speed it is played. If the download speed matches or exceeds the rate at which the audio data is played, the audio can be played indefinitely without any interruptions. There are many practical applications for this technology on the Internet, and numerous commercial applications in the streaming audio client/server market have emerged. In this article, I'll present idtAudio, a streaming audio applet written in Java. (The source code for idtAudio and the companion program, idtAudio.cpp, is available electronically; see "Resource Center," page 3.)

As a Java applet, idtAudio inherits several advantages over most other streaming audio tools. Its most distinct advantage is that users never need to install any software. As an applet, the code is downloaded automatically into the browser as needed without user intervention. Furthermore, since the code is downloaded for each audio instance, there is no possibility of version mismatches between the audio data and the player. Also, applets are portable. The idtAudio applet seems to run equally well on Netscape Navigator 3.01 and Microsoft Internet Explorer 3.01 on Windows 95/NT.

The design of idtAudio also sports several distinctive features. The streaming protocol is built on standard HTTP, which is the same protocol used by all web browsers. As such, the applet can download its audio data through any firewall that allows web activity. This also means that the server has no requirements for special drivers or extensions.

Compression

Compression is the key to transmitting streaming audio data at low bandwidths. Audio data is considered to be one of the most difficult mediums to compress, but when using Java, the algorithm must also be very fast.

I researched several existing audio compression technologies, starting with MPEG. While its audio quality is excellent and several code samples are readily available, MPEG's psychoacoustic algorithm places a heavy toll on the CPU and cannot be implemented effectively in Java.

ADPCM is a simpler algorithm that would seem more likely to work in the Java environment. Unfortunately, ADPCM does not easily compress to the levels required for streaming modem transmission. The quality becomes poor, and the specific algorithms used become more secretive.

Other algorithms had similar problems, so I was forced to develop my own. Because compression relies on predictable patterns to be effective, I began by studying graphs of wave forms (see Figure 1). The sinusoidal qualities of the wave suggested a first algorithm: Compress by capturing the peaks and valleys only, and decompress by programmatically filling in the remaining points using a sinusoidal function. With this algorithm, the decompressed audio had an annoying echo effect. More importantly, I also realized that the algorithm was uncompromising in that it could never predictably compress to any particular ratio. For example, a wave composed entirely of peaks and valleys would double in size after being "compressed."

A common technique used in modern physics is "curve fitting," where empirical data is approximated to a mathematical function such as a line or a parabola. My next attempt at audio compression broke up the wave form into a series of straight lines that approximate the original. The audio quality was much improved, and the compression ratio could be controlled by how closely the original wave form was followed. This tolerance level could be iteratively increased until the desired compression was achieved. The only problem was that, to achieve the required compression ratios, the tolerances had to be set very high. This degraded the audio quality too much. I still needed a way to increase the compression without sacrificing sound quality.

A frequency analysis of the audio data showed that smaller values occur much more regularly than larger ones, so it seems natural to sacrifice precision at higher amplitudes and preserve more detail near the smaller amplitudes. Originally, each audio sample is a signed eight-bit byte. idtAudio further compresses the audio data by storing a four-bit value for the amplitude instead of the full eight bits. The four-bit signed value is the square root of the eight-bit value (with negative amplitudes represented by negative square roots). Right or wrong, this extra 33 percent of compression was enough to make idtAudio work.

Although the audio compression program, encode.exe, was originally written in Java, performance was poor on large files, so I rewrote it in C. The C version seems quick enough for streaming encoding. Thus, we have laid the foundations for an Internet telephone, but that would be a different article entirely...

Decoding and Playing the Compressed Audio

Due to Java's poor audio support, the audio player applet requires the use of Sun's unsupported *AudioPlayer* class in the *sun.audio* class library. (It is this dependency that prevents idtAudio from working with every Java-capable browser.) The *AudioPlayer* class requires an *InputStream* object from which it reads and plays ulaw-encoded audio data until it reaches the end of file. The idtAudio applet derives its own class from *InputStream* called *FifoInputStream* (which will be referred to as FIS for reasons I'll explain shortly), and submits it to the *AudioPlayer* class. FIS allows one write thread and one read thread to communicate with fairly good performance.

A simple architectural solution, then, would have one thread downloading, decoding, and depositing audio data into the

---

Rick is a System Engineer for Intel Corp. He manages a central support group in a department that is responsible for NT and UNIX infrastructure design and integration with enterprise applications. He can be contacted at rmoore@primenet.com.

FIS queue, while another thread (implemented via *AudioPlayer*) would read and play the audio data; see Figure 2. This design worked well when the compressed audio data was read from a local hard disk. Streaming from a true Internet URL was a different story. The audio was broken up and my modem lights indicated a lack of activity. Some quick tests of URL download times with C and Java programs suggested that the server's performance was not an issue, and that the Java language, itself, was not an issue. Clearly, the simple design described earlier was not an optimal solution.

Because a serial download leaves the CPU mostly idle, it made sense to break up the download and decode tasks into separate threads; see Figure 3. The FIS class was the perfect way for the two threads to communicate. A *DoT* (short for "download thread") class, derived from *Thread,* would be responsible for downloading the audio data and depositing it into the first FIS object. The *DeT* (short for "decode thread") class, also derived from *Thread,* would be responsible for reading the first FIS object, decoding the data, and depositing the results into the second FIS. The *AudioPlayer* class would then read and play the second FIS at its leisure. This solution proved to be both successful and elegant.

Optimizing Java Code

Some serious optimization was still required to make the player perform on a 486 running Netscape's slightly slower implementation of Java.

First, the synchronized sections of code were minimized. Some basic common sense pays off here. Consider the nature of our FIFO queue: One thread reads at the tail and another writes to the head. The only point of contention is the full member that indicates how full the queue is. Write operations to this member are synchronized. Read operations are not synchronized: This does carry a slight risk, but the risk only occurs under an unusual set of circumstances and is considered acceptable for the performance gain.

Every time an array is accessed in Java, the language performs some internal bounds checking. This checking can be costly. Not only was array access minimized, but looping itself was also minimized. Buffered access to the FIS objects was encouraged wherever possible in favor of character access.

Probably the most significant performance boost came from the *System.arraycopy()* function, which copies one array's contents to another, given offsets and lengths into each. *System.arraycopy()* is analogous to *memcpy()* in C and similar in performance. Sadly, the only alternative in Java is to use a standard loop and iteratively copy one member at a time.

Instead of a circular FIFO, I considered using a linked list of dynamically allocated buffers. When data was entered into a FIFO, a new buffer would be allocated and linked to the head of the list. Reading would retrieve and unlink tail-end buffers. Such a scheme could avoid one of two calls to *System.arraycopy()*, but would add the expense of garbage collection and object creation. A couple of test programs showed that *System.arraycopy()* was still slightly faster than the linked list scheme, so I never implemented the change.

Information on Java optimization is sketchy at best. Most texts suggest the generic approach of optimizing the innermost loops. The tricks I've just described showed remarkable performance gains, however. These techniques can be applied to almost any program.

Protecting Java Code

You may wonder why I chose such cryptic naming conventions for my classes. Originally, they were called *FifoInputStream*, *DecodeThread*, and *DownloadThread*, but the destiny of my applet was uncertain at that time. Because Java classes are downloaded by name, anybody using the audio player could quickly discover some key design secrets! The names were too informative, so I renamed them to *FIS*, *DeT*, and *Dot*, names that would be meaningless to anyone who didn't already understand the applet's internal architecture.

Another protection mechanism demonstrated by idtAudio is the "time-bomb" feature often seen in shareware programs. (The code is commented out to effectively disable the time-bomb feature, but remains for your reference.) Typically, an expiring license scheme is not very effective in Java applets because a web developer could easily download a licensed copy of the applet from another web site. The trick to making this work is to place the "bomb" in the audio data instead of the applet itself. By doing so, the key to the license is embedded in data that is useless to everyone but the web site owner who owns the license. This is accomplished rather simply by placing a time stamp within the header information. The applet checks the time stamp and refuses to play encoded files that are too old. When registered, however, the encoder inserts null data into the time stamp slot, which indicates that the encoded audio file never expires.

Using idtAudio

The first step to using the idtAudio applet is to create your audio files. I use Syntrillium's Cool Edit 96 to create mine; even the disabled shareware version has enough features to do the job. Record an audio session and save it as an 8000 MHz eight-bit sam file.

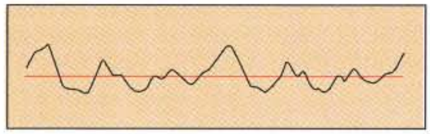


Figure 1: *Graph of a typical audio wave.*

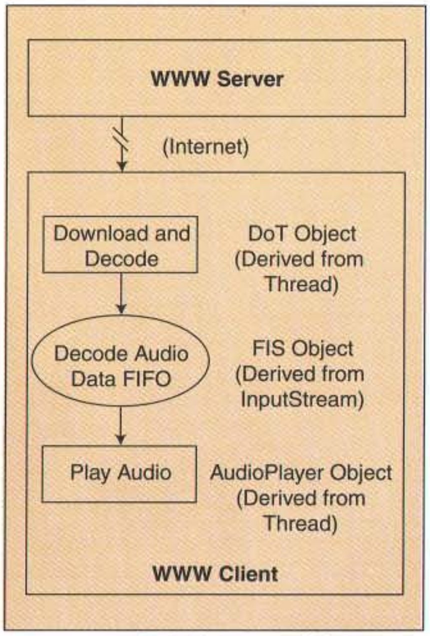


Figure 2: *Block diagram of first applet algorithm.*

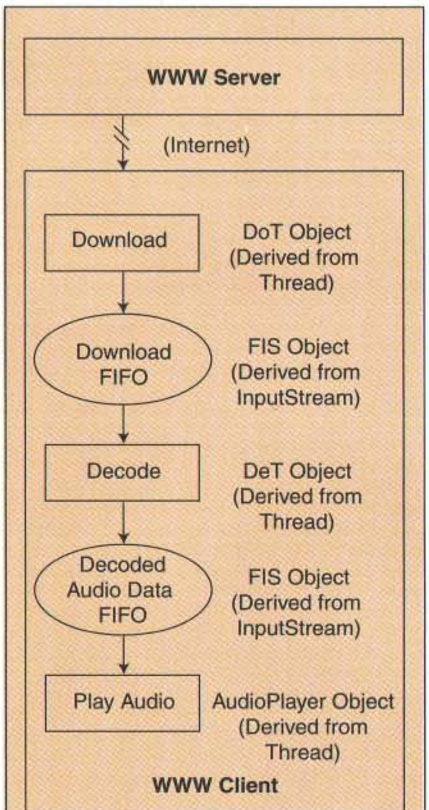
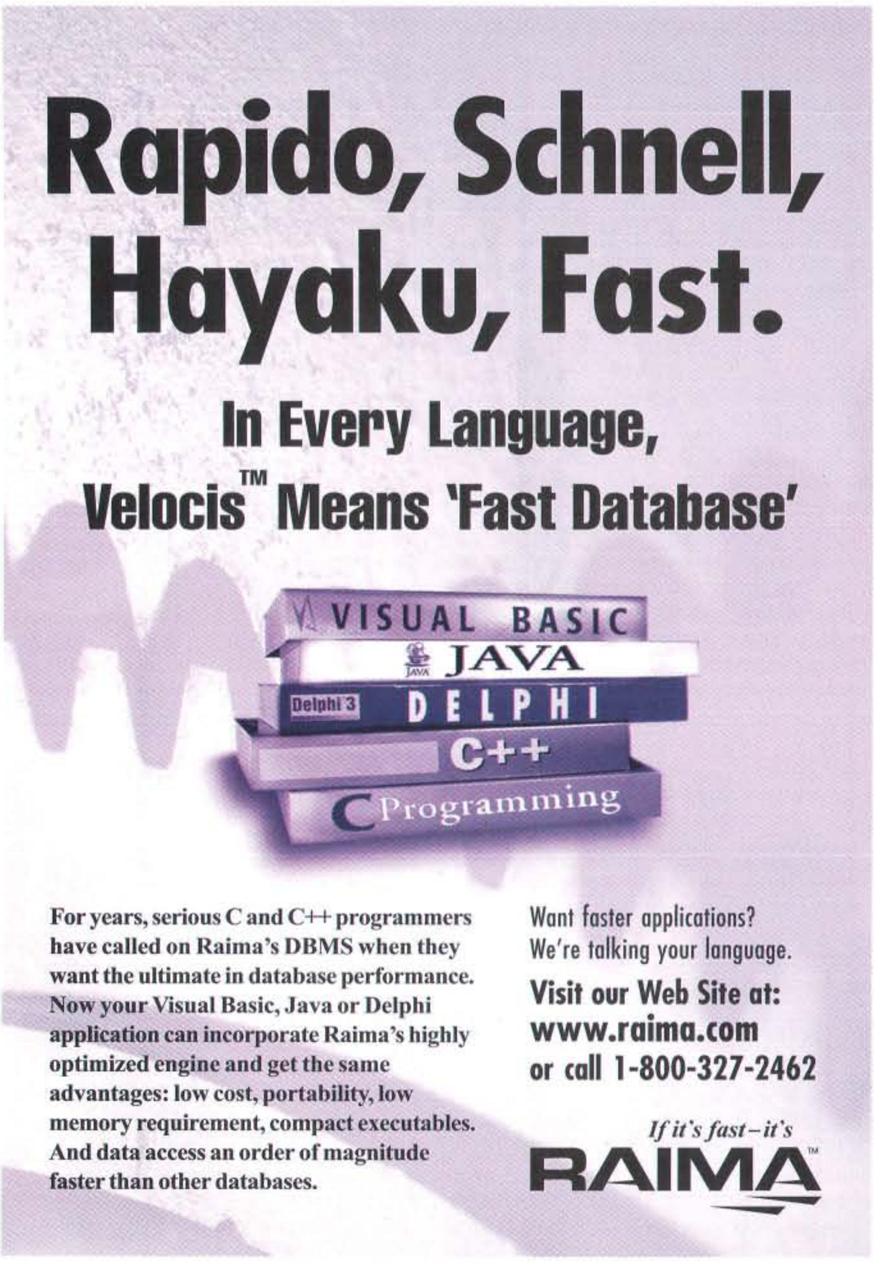


Figure 3: *Block diagram of second and final applet algorithm.*

Next, the audio file needs to be compressed. Use the encode.exe program for this. It requires an input file and an output file for command-line parameters. An optional third parameter is either a tolerance level or a target bandwidth (in characters per second), depending upon the value given. If a tolerance is given, the audio file is compressed once at the specified tolerance. If a target bandwidth is specified, the audio file is compressed iteratively at increasing tolerances until the target bandwidth is reached.

```
<applet code="idtAudio.class" align="baseline" width="80" height="30" id="idtAudio">
<param name="AudioStreamURL" value="fish.idt">
<param name="AutoPlay" value="false">
<param name="Debug" value="true">
</applet>
```

***Example 1:** Adding the idtAudio applet.*



My experiments suggest that a 28.8 modem connected at 24000 bps can sustain transmission speeds of approximately 2300 cps, so this is the default target bandwidth when none is specified. A 14.4 modem can only sustain bandwidths of about 1400 cps. Some experimentation has uncovered a rule of thumb: Under good conditions with PPP, a modem's bandwidth in characters per second is approximately the modem's connection speed in bits per second divided by 10. To be safe, I usually subtract another 100 cps.

After the encoded audio file is created, it must be stored on your web server. You may wish to use a familiar MIME type if your server is picky about such things. The four applet classes for the player (*DeT*, *DoT*, *FIS*, and *idtAudio*) must also be stored on the server. Add Example 1 to your HTML code to add the idtAudio applet.

In this example, "fish.idt" would be the name of the encoded audio file. The *Debug* parameter sets the style of the applet's appearance; *true* provides more detailed information. The *AutoPlay* parameter indicates whether the applet should begin playing immediately or wait for the user to click on it.

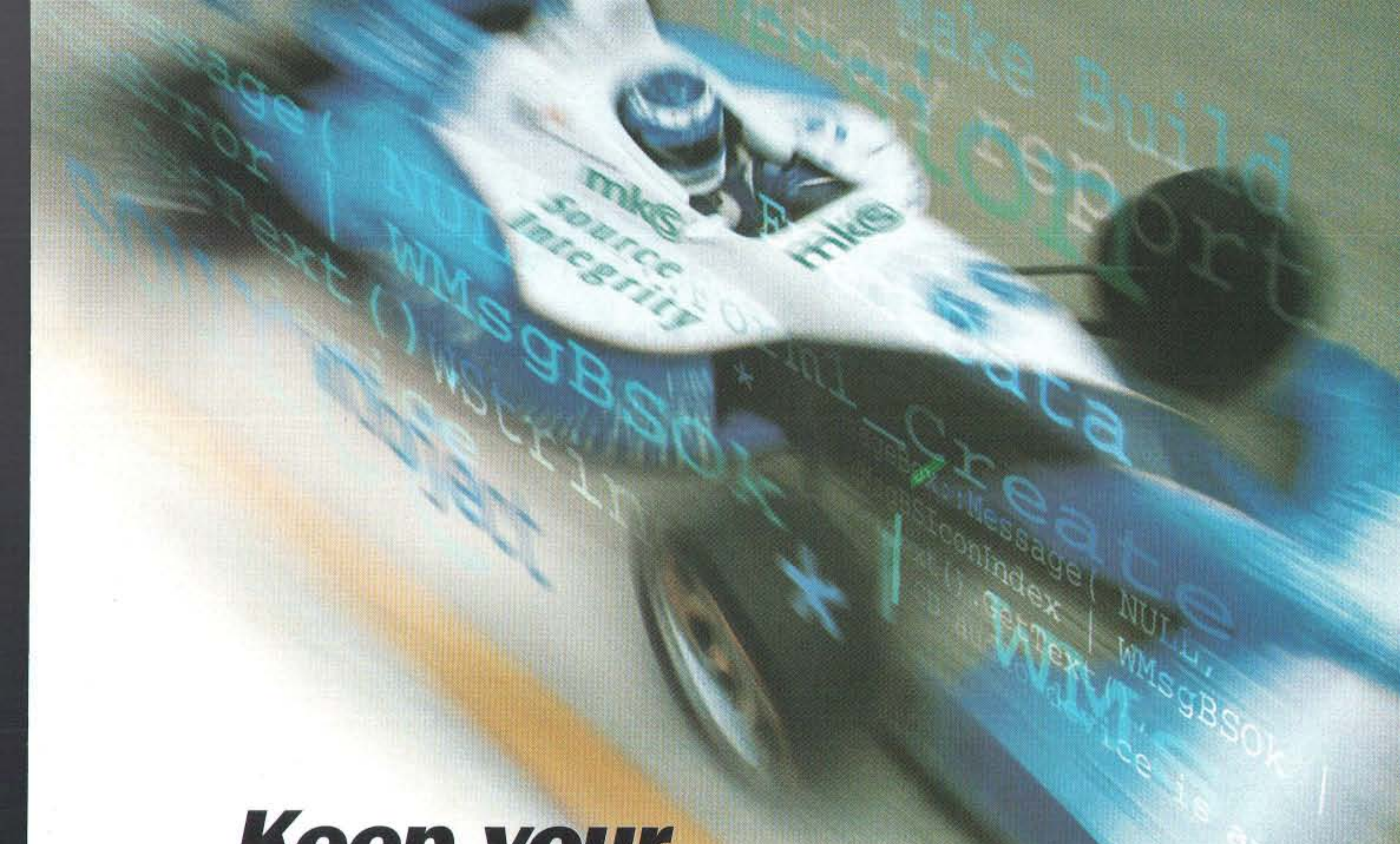
Conclusion

In retrospect, there are a number of variations to the compression algorithm that could increase compression and quality. You could even implement an "algorithm shifting" scheme that would change compression algorithms on the fly based on an analysis of the audio data.

Simpler improvements would be to take the amplitude compression into account when calculating tolerances. The amplitude compression introduces substantial error that is not considered by the tolerance level. Also, the amplitudes might be better stored as offsets instead of absolutes, similar to the ADPCM method.

Nevertheless, the idtAudio applet exemplifies a number of concepts. First, it is a good example of object-oriented programming. In fact, I believe every object-oriented programming principle is demonstrated in some way within the applet's code. Next, it demonstrates the power of Java. Consider that the total code size is only 17 KB yet it implements a multithreaded graphical applet with performance good enough to perform downloading, number crunching, and real-time audio on a 486 machine. Lastly, it demonstrates a basic security and licensing scheme that developers may want to apply to commercial applets.

DDJ



Keep your development team on track.

Speed your time to market with MKS Source Integrity,™ the highest performance SCM solution.

Only MKS Source Integrity offers the complete cross-platform Software Configuration Management (SCM) solution you need to keep your winning development team on track—driving full speed toward your client/server and Intranet application development objectives.

MKS Source Integrity gives you the award-winning combination of powerful process control and parallel development support—everything your team needs to ensure quicker time to market for products. Plus, you'll have an easy-to-use solution that includes advanced security, the best integrations into the most popular IDEs, reporting, and powerful project oriented release management for unbeatable SCM performance and total project control.

Already established as a front-runner in its own right, MKS Source Integrity, from the makers of MKS Toolkit, is the high-performance SCM solution, the acknowledged choice of software development professionals worldwide!

For your free test drive, call 1-800-613-7535

www.mks.com

mks

International Headquarters MKS UK Limited: +44 (0) 181 335 5920 • MKS France: +331 (0) 3082 2762 • MKS Germany: +49 (0) 711 3517750

MKS, MKS logo, MKS Source Integrity, and MKS Toolkit are trademarks of Mortice Kern Systems Inc. All other trademarks acknowledged. © 1997.

Sorting and Searching Linked Lists in Java

John Boyer

Generic collection classes, such as those in the C++ Standard Template Library and Java Developer's Kit, are nice to have around, in part because they encapsulate standard algorithms that are useful in many situations. But such libraries are hard to write—how do you know, for instance, that the particular algorithm you chose will work well across so many applications? As you'll see upon reading John's article, even experts such as those writing the JDK can sometimes overlook the best algorithm choices.

I found John's discussion of linked list searching especially interesting. Like many people, I failed to distinguish between the cost of walking a link and the cost of a key comparison. Understanding that distinction makes a big difference.

—Tim Kientzle

Some common misconceptions in computing are that arrays can be sorted more quickly than linked lists; that the quicksort is the fastest comparison sort on average; and that binary searching a linked list provides no benefit over a sequential search because both are *O(n)*.

If you agree with any of the aforementioned views, you're in good company. The list sort and search algorithms in the recently released Java Developer's Kit (JDK) 1.2 beta 2 are based on these misconceptions, which results in suboptimal performance.

In this article, I'll present Java implementations (available electronically, see "Resource Center," page 3) of more-efficient list sort and search algorithms, which are based, in part, on my undergraduate thesis work in 1988. These algorithms also have the advantage that they work on both singly and doubly linked lists (although the sorts require a simple post-processing loop for doubly linked lists to reconnect all of the references to previous nodes).

Performance

Although I collected empirical results on a 64-MB 200-MHz Pentium running Windows 95, I also ran the tests on other machines. While the numbers change, the conclusions are the same.

The test program allows you to specify the number of randomly generated ten-character strings to sort or search. As you type your number, the program will not show you what you are typing. This is a bug in Java's *Kbd.readLine()* function. However, if you simply type your number on faith and press Enter, your input will be properly recognized. You can then choose a sort or search test. The sort test runs five different sort functions: the array quicksort in *Arrays.sort()*, the linked-list quicksort in *Collections.sort()*, a new linked-list quicksort, a new linked-list merge sort, and a nonrecursive version of the linked-list merge sort. The search test compares the new binary search to the sequential search in *Collections.binarySearch()*. The search tests simply use the search to find every string in the array or list.

Figure 1 shows the results of the sort tests on 16,000 strings. The limitations of *System.currentTimeMillis()* (which is only updated every 50 to 60 milliseconds on Windows 95) make it difficult to distinguish the two list merge sorts. With 32,000 strings, the Java VM froze without any errors, so I was unable to empirically test which of my two new list sorts was faster.

All of the new linked-list functions sort linked lists faster than *Arrays.sort()* can sort the equivalent array. Since *Collections.sort()* calls *Arrays.sort()*, naturally all of the new linked-list sorts are much faster than the JDK's linked-list sort. This was not just on average but in every test case. The most notable result is that the JDKv1.2b2 *Collections.sort()* was 33 percent slower than the recursive merge sort, and the *Arrays.sort()* function was slower than the linked-list merge sort by 22 percent. Best of all, the merge sort cannot degrade to quadratic time for any data set. Thus, it is not true that linked-list sorting must be slower than array sorting, nor is it true that the quicksort is the fastest comparison sort.

As Figure 2 shows, the linked-list binary search was over nine times faster on average than the sequential search using 1000 to 4000 randomly generated strings. The shapes for both searches are parabolic because the searches are *O(n)*, and the test program searches for every string in the list, yielding $O(n^2)$ test performance.

Small Errors, Big Impact

When first testing *Collections.sort()*, I noticed it was getting results that were poorer than I expected—140 seconds for 16,000 strings. *Collections.sort()* only adds a few linear time operations to an *O(n log n)* array sort, so it shouldn't be much slower than array sorting.

The first step of *Collections.sort()* is to convert the linked list to an array; this was taking 136 seconds in my first test. The *toArray()* function itself turned out to be $O(n^2)$, apparently because the iterator's *next()* function in *LinkedList* restarted at the beginning of the list in each call. As

John is a software-development manager at UWI.Com—The Internet Forms Company. John is also a part-time theoretical computer-science graduate student at the University of Victoria. He can be contacted at jboyer@uwi.com.

a temporary fix, my test program subclasses *LinkedList* to overload the *iterator()* function. The new version merely returns *listIterator()*, with a substantial performance improvement.

Linked-list Quicksort

It may come as something of a surprise that the quicksort can be done on a singly linked list since it is impossible to efficiently traverse backwards. In fact, I haven't encountered any other work describing a linked-list partition sort, so I believe that this is an original algorithm (though I'd be glad to hear from any reader who knows otherwise). Listing One (listings begin on page 137) is Java source for this new sort.

The basic method uses the first node's key as the pivot. The algorithm traverses forward through the list using two references called *aNode* and *aNodePrev*. Nodes with a key value greater than or equal to the pivot are ignored. When a node containing a lesser key is encountered, the algorithm deletes *aNode* by connecting *aNodePrev.next* to *aNode.next*. Then, *aNode* is pushed onto the front of the list, becoming the new first node. Once the list is divided into two sublists, it can be sorted by recursively applying this partitioning procedure to each sublist.

This still leaves the quicksort's problem of quadratic performance on sorted/nearly sorted and reversed/nearly reversed data. The simplest and most popular solution to this problem is the median of three method, which can be effected in constant time per subarray partition. However, the median of three method requires linear time to scan the entire linked list prior to the partition pass. Truthfully, this really isn't such a bad idea. Since the partition already takes linear time, having the median of three method take $O(n)$ time doesn't change the $O(n \log n)$ average case rating of the sort. The demo program (available electronically) provides a median of three function for the list quicksort.

But if linear time is acceptable, then it is possible to do something more elegant. In the case of the linked-list quicksort, the pivot advancement method (see Figure 3) is much shorter to code and easier to understand, yet it results in much faster performance on random data as well as $O(n)$, overall sort behavior on nearly sorted and nearly reversed lists.

Pivot advancement starts with two references, *Pivot* and *aNode*, at the start of the list. The algorithm advances *aNode* through the list as long as it is monotonic nondecreasing, and every two advancements of *aNode* yields one advancement for the *Pivot*. If the pivot advancement loop traverses the entire sublist, then it is sorted so the function returns with no further processing. If the pivot advancement loop does not traverse the entire loop, then the pivot is the median element of the entire monotonic nondecreasing sublist at the start of the input list. Furthermore, we leave *aNode* at its given position rather than starting the partition loop at the node after the pivot because the nodes between the *Pivot* node and *aNode* are already known not to have lesser key values than the pivot.

The quicksort is not the quickest comparison sort. The merge sort is

Thus, with only one extra comparison per partition, pivot advancement handles nearly sorted data sets in linear time rather than quadratic time. Reversed data is also handled in linear time because nodes pulled from the second sublist are pushed onto the front of the first sublist, which reverses them into sorted order. On random data, pivot advancement outshines median of three because the former does little work in choosing a pivot at or near the beginning of the list (rather than traversing the whole list). The probability that *aNode* will be advanced in random data sets diminishes exponentially (1 in 2^i chances of i advancements).

Finally, note that the linked-list quicksort presented here still has a quadratic worst case. The worst case occurs with sequences with the pattern 3,2,1,6,5,4,9,8,7... However, these worst case examples are unlikely to occur in practice, and the expected running time is $O(n \log n)$ with low hidden constants.

Linked-List Merge Sort

Conventional wisdom has it that the quicksort is quicker than all other comparison sorts, and the merge sort in particular. However, this conclusion is generally based on analyzing the merge sort's performance on an array, which is not the ideal data structure for the merge sort. The merge of two sorted arrays requires an auxiliary array, which ultimately results in the merge sort performing twice as many movement operations as the quicksort on average. By comparison, the linked-list merge operation can be done in-place with only constant memory overhead for a few temporary variables. This eliminates both of the detracting factors for the merge sort. In fact, since the merge sort performs fewer key comparisons than the quicksort on average, it is the merge sort that should be quicker when sorting linked lists.

As the empirical results show, the linked-list merge sort (see Listing Two) on a linked list is so fast that it outperforms the quicksort in *Arrays.sort()* using the same data placed in an array. The list quicksort presented in this article is also slower than the list merge sort by an average 6 percent. Furthermore, the merge sort's recursive calls always divide the list evenly in half. So, given that the merge step is a simple linear time process, the merge sort can never be slower than $O(n \log n)$. Finally, the guaranteed logarithmic function call depth means that the merge sort will never generate a stack overflow exception. A worst case array quicksort could generate an exception in Java with as few as 5000 elements, which is the current size limit on Java's function call stack.

Nonrecursive Merge Sort

It is also commonly believed that nonrecursive sorts are better than recursive ones. Again, this is due to the quicksort. It is better to use a nonrecursive quicksort simply to guarantee that no stack overflow exception will occur if a worst-case data set is encountered. The merge sort doesn't have this problem, so recursion poses no threat. However, a nonrecursive version does avoid function calls, so I coded two nonrecursive merge sorts. To my surprise, neither of the nonrecursive solutions managed to outperform recursive merge sort.

The first nonrecursive method traverses the list merging all lists of sizes 2, then 4, 8, and so on. It properly handles uneven cases with a minimum of overhead (see Listing Three). The second method is slightly faster, so its times are the ones reported in Figure 1. It is much more complicated. The idea was inspired by the consolidation step of the Fibonacci heap. The idea is to build an array of lists, all initially empty. Each list is responsible for holding the number of nodes in successive powers of 2. You add the nodes from the original list one at a time. Then you add nodes, one at a

Dr. Dobb's JOURNAL technetcast
audio + video + chat events for developers
ddj@technetcast.com www.technetcast.com

Upcoming Programs: send us advance questions — audio + video + chat

Y2K and other thoughts on the Millennium .... with Bob Bemer
Code Complete .... with Steve McConnell
OO Programming in C++/Java .... with Bruce Eckel
ActiveX/COM .... with Richard Hale Shaw

Regular AudioColumns

Radio Flames by Mike Swaine — bi-weekly, audio + text
Silicon Valley Chronicles by Jean-Louis Gassée — weekly, text

Archives

All past shows available on the web site: C++, C, DHTML, ActiveX, Perl, Winsock, Metrowerks, VRML, MFC *and more.*

Special Events

Visit TechNetcast for live coverage of developer events. Check out our reports from SD '98, Be DC and JavaOne, including live interviews with Bjarne Stroustrup and James Gosling.

host: philippe lourier
email: ddj@technetcast.com

TechNetcast is produced by Dr. Dobb's Journal and the Pseudo Online Network (www.pseudo.com)

time, to the 2^0 list. That list overflows on the second addition, at which point it is moved to the 2^1 list, emptying the 2^0 list. Then, if you add another two nodes, the 2^0 list overflows again. But this time you have to merge it with the existing 2^1 list. When done, the 2^1 list has four elements, so it is moved to the 2^2 list. The procedure is conceptually similar to incrementing a binary number. If there were 15 nodes in total, then the first four lists (2^0, 2^1, 2^2, and 2^3) would be full. If you then add a 16th node, then successive merge operations would join all nodes into the 2^4 list, emptying all other lists in the process.

Though more elaborate, this method is faster than the first nonrecursive merge sort because it does not traverse any sublist twice. I also built a second merge function because this sort didn't need some of the parameters and extra work of the recursive sort's merge function. Despite this, the second nonrecursive merge sort did not run faster than the recursive merge sort. The empirical results in Figure 1 show the nonrecursive

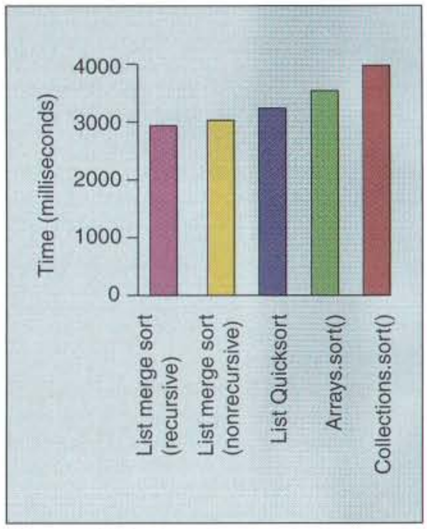


Figure 1: *Sort comparison.*

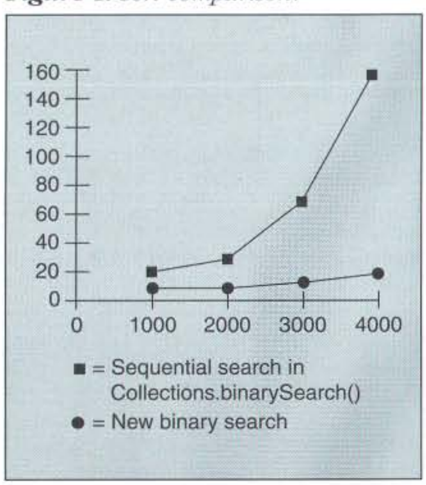


Figure 2: *Search comparison.*

TCP/IP • SMTP • UDP • HTTP • POP3 • BASE64 • MIME • UUENCODE • FTP • T.30 • T.70NL • X.75 • HDLC • CAPI • ISDN • HDLC • ZMODEM • XMODEM • WAV • VOICE • FAX • ASYNC • WHOIS • FINGER • SOCKET • SMSC • UCP • TAP • 3964 • 3964R • GATEWAY • HDLC • SDLC • BSC • FRAME RELAY • CUSTOM PROTOCOLS • CUSTOM DRIVERS • SFF • TRACE • TCP/IP • SMTP • UDP • HTTP • POP3 • BASE64 • MIME • UUENCODE • FTP • SOCKET • FINGER •

LUCA SOLVES THE PROTOCOL PUZZLE

LUCA integrates all popular transmission media, protocols and services into one comprehensive software development framework. Take control of the Internet with ease. Interconnect different protocols and transport media. For example, route received email to pagers. Share code for different transmission media (modems/ISDN etc.). And complement the wealth of LUCA's standard product components with custom protocols and drivers.

LUCA API + CONTROLS
PROTOCOL LAYER(S)
TCP/IP ASYNC ISDN OTHER

- integrates the Internet, fax, voice, pagers, and more
- C++, Delphi, Visual Basic, PowerBuilder, Java
- modular, scaleable, expandable

FREE EVALUATION CD 1-800-986-6578

In the USA contact:
Griffin Technologies
Tel. 785-832-2070
Fax 785-832-8787
www.griftech.com

In Europe contact:
Langner GmbH
Tel. +49-40-27 10 06
Fax +49-40-280 75 57
www.langner.com

merge sort to be 0.3 percent slower than the recursive version, but it is really too close to call.

Sort Stability

The merge sort provides stability, a feature that the quicksort doesn't provide without extra programming (including the array sort in the JDK). Stability in this case refers to a sort's ability to keep equivalent keys in the same order that they appeared in the unsorted data. The test program has a few lines of commented code that can be uncommented to demonstrate stability. The test simply adds the field *Pos* to each node. Before sorting, the numeric position of each node is recorded. After the sort, the key data and *Pos* value of each node is printed. After the merge sort, the position values for equivalent keys are always ascending, but they are in random orders for the list quicksort. To make a stable quicksort, you could include *Pos* as a secondary key, but this slows the sort and requires an extra field in the node structure. With the merge sort, stability is free.

Binary Search

According to the empirical results, the linked-list binary search (Listing Four) was the biggest winner. While it's true that a binary search requires sorted order, this is often required anyway. And even when it's not, the results show that, in Java, the cost of sorting is paid for with relatively few searches (for example, about 25 searches on a 4000 element list).

The algorithm starts by setting *PartitionFirst* equal to the first list element and *PartitionSize* equal to the number of list nodes. In each iteration of the outer loop, the inner loop traverses to the list's middle node. If the search key is less than the key at the middle node, then the search should continue in the first half of the list. So, the algorithm merely reduces the *PartitionSize* because the *PartitionFirst* already refers to the start of the desired sublist. If the search key is greater than the middle node's key, then *PartitionFirst* is set equal to the node after the middle node, and the *PartitionSize* is set equal to the count of nodes after the middle node.

On average, the sequential search does $n/2$ node hops and $n/2$ comparisons. The number of node hops in a binary search is never less than $n/2$ but is almost always at or near n. Despite the fact that the binary search does nearly twice as many node hops, the binary search is over nine times faster than a sequential search because the binary search does only $log(n)$ comparisons. The linked-list binary search pays for itself if key comparisons are more complicated than two or three node hops. This is true of many data sets—for example, search by name requires string comparisons—but key comparisons are especially costly in Java because *compareTo()* is a virtual function invocation.

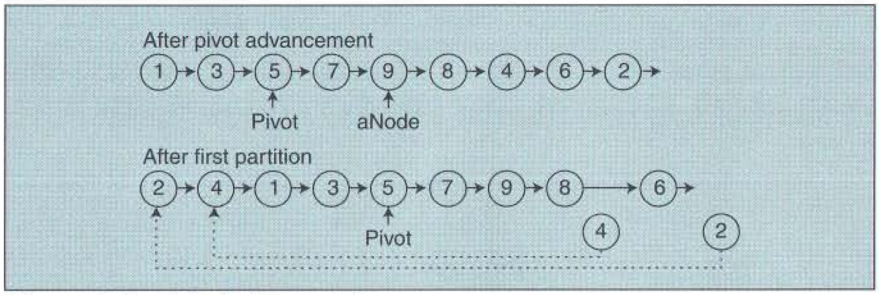


Figure 3: *Singly linked list quicksort.*

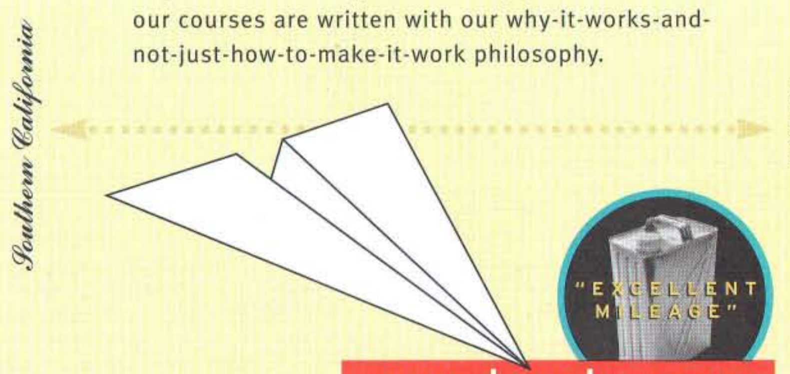
DDJ

(Listings begin on page 137.)

COM·DCOM·OLE·NT SERVICES
hands-on developer training courses:
ACTIVEX · VISUAL BASIC · DDK
C++ · JAVA · WIN32 · SECURITY

OPEN ENROLLMENT • ON-SITE
GUERRILLA EVENTS • WORKSHOPS

Developers learn best from other developers. Regardless of the type of training you choose, all our courses are written with our why-it-works-and-not-just-how-to-make-it-work philosophy.



www.develop.com
REGISTRATION AND INFORMATION:
310.214-7800

TRAINING AND EDUCATION FOR DEVELOPERS LIVING THE COM LIFESTYLE
DEVELOPMENTOR
3547 VOYAGER STREET, SUITE 201, TORRANCE, CA 90503

AD LINK 388

VME: Coming Out of the Cold

Robert R. Collins

In the first two installments of this series on Intel's Virtual Mode Extensions (VME), I examined why VME is needed. More specifically, in the January column I explained the performance problems associated with Intel's original v86 mode implementation. I went on to discuss Intel's new Enhanced v86 Mode implementation, in particular, how Ev86 mode fixed many of the performance deficiencies in the first v86-mode implementation.

In the March installment, I went under the hood of VME, examining all of the various components which are used by the VME, and explained how they work. At the end of that column, I presented source code that can be used to model your own Ev86 mode tasks.

Subsequent to those columns being published, Intel released the *Intel Architecture Software Developer's Manual, Volume 3*. Although six months late, the manual couldn't have been released at a better time because it gave me the opportunity to determine whether Intel had finally documented the missing details of VME or described its many caveats.

Alas, Intel has done neither. The missing details are still missing, and the caveats of VME have never been described. Notwithstanding these omissions, Intel has actually removed some vital VME information.

This month, among other things, I'll show how many references to VMEs remained in the Pentium manuals after Intel's lawyers had forced their removal. These oversights allowed the Appendix H liberation movement to reverse engineer the VME details and release the information six months before Intel.

The Appendix H Liberation Movement

After the creation of the notorious Appendix H, you would have thought Intel had removed every reference to the "secret" VMEs (see my November 1997 column for a description of Appendix H). However, Intel wasn't careful enough. In fact, I found many references to VME scattered throughout the Pentium manuals.

When preparing to help reverse engineer the VME details, I started with the premise that the Pentium manual was virtually silent on the subject of VME. In my research, I found many more references than expected. I extracted all the references and began categorizing them. As I rearranged the quotes into categories, I found the Pentium manual told its own VME story. This story starts by listing some of the problems with the existing v86 mode. Next, the story establishes that new VMEs were implemented to enhance performance. By the end of the story, the manual tells you virtually every aspect of VME. From this information, any capably equipped engineer can write code to figure out the various implementation details.

Here's the Pentium manual's own story of why VME was needed, and how to use it. Unless otherwise noted, all of the following quotes are taken from the *Pentium Processor Family Developer's Manual, Volume 3: Architecture and Programming Manual* (Intel part number 241430).

- "Many 8086 programs written for nonmultitasking systems set and clear the IF flag to control interrupts. This may cause problems in a multitasking environment. As a result, virtual monitors running on the Intel386 and Intel486 processors require maintaining a virtual interrupt flag in software. All instructions affecting the IF flag trap to the virtual-8086 monitor for emulation on these processors." (Section 22.5.) Trapping to the v86 monitor causes considerable microprocessor time to be spent in the overhead of faulting and returning control to the program under emulation.
- "The Pentium Processor includes extensions to its virtual-8086 mode of operation that improve the performance of applications by eliminating the overhead of faulting to a virtual-8086 monitor for emulation of certain operations." (Chapter 22.)
- "By enabling the virtual mode extensions, the virtual-8086 mode performance of the Pentium processor is significantly improved." (Section 22.10.)
- "Existing interrupt flag sensitive instructions provide significant performance improvement when using the virtual mode extensions of the Pentium processor." (Section 22.1.1.)
- The interrupt flag sensitive instructions which "eliminate the overhead of faulting to a virtual-8086 monitor for emulation" are CLI, STI, PUSHF, POPF, INT *n*, IRET. (Section 10.1.3.)
- "When IOPL=3, interrupts are serviced by the protected-mode interrupt service routine in a manner compatible with the Intel486 processor. On the Intel386 and Intel486 processors, all INT *n* instructions running in virtual-8086 mode require interception by the virtual-8086 monitor when IOPL is less than 3. For information on Pentium processor virtual mode extension support of interrupt handling, see Appendix H." (Section 22.6.) This statement is very significant because it essentially tells the reader that INT-*n* instructions running in Pentium Ev86 mode aren't required to be intercepted by the virtual-8086 monitor.
- Setting CR4.VME=1 "enables support for a virtual interrupt flag in virtual-8086 mode." (Section 10.1.3, Section 22.10.)
- "The Pentium processor TSS contains additional information used in virtual-8086 mode by the virtual mode extensions to the Pentium processor." (Section 23.2.17.1.)
- "The fields of a TSS are divided into two main categories:…(We are only concerned with the second category.) 2. Static fields the processor reads, but does not change. These fields are set

Robert is an independent consultant on the x86 architecture. He can be reached at rcollins@x86.org.

up when a task is created. These fields store:...The base address for the I/O permission bit map and interrupt bit map. The base address points to the beginning of the I/O map and the end of the 32-byte interrupt map...See Chapter 22 for more information about the interrupt redirection." (Section 13.1.) This statement clearly tells us that there is a 32-byte interrupt redirection bit map stored in the TSS, whose tail is pointed to by the I/O permission bit map base address.

- The EFLAGS register contains two flag bits (VIF, VIP) used by VME. (Section 3.3.4.1, Section 10.1.1, Section 23.2.9, Section 23.2.9.1, Instruction description for IRET and POPF.)
- VIF and VIP will always be clear on processors that don't support VME. (Intel source code for CPUID detection.)
- "The VIP flag together with the VIF enable each applications program in a multitasking environment to have virtualized versions of the system's IF flag." (Section 10.1.1.)
- "The VIF is a virtual image of IF (the interrupt flag) used with VIP." (Section 10.1.1.)
- CLI, STI, PUSHF, POPF, INT-n, and IRET are all affected by enabling VME. (Instruction description for CLI, STI, PUSHF, POPF, INT-*n* and IRET.)
- IRETD or POPFD never modifies VIF and VIP. (Instruction description for IRET and POPF.)
- The VIF bit "operates between a STI state for enabling the execution of interrupt instructions initiated by [a] previously written program and a CLI state for disabling the execution of [the] interrupt instructions." (UK Patent Application GB 2 259 794 A, Claim 1.a.i.)
- The VIP bit "operates between a pending state during which interrupt requests are awaiting execution and a nonpending state in which no such interrupt requests are awaiting execution." (UK Patent Application GB 2 259 794 A, Claim 1.a.ii.)
- The Pentium contains the means to let the operating system "(change) the state of the VIF bit of the EFLAGS register without the use of emulation software, so long as the VIP bit is in its nonpending state." In other words, when using Ev86 mode, CLI and STI modify VIF, without blocking or enabling external interrupts, and without modifying the actual interrupt flag (IF). (UK Patent Application GB 2 259 794 A, Claim 1.b.)
- Once VIP is set into EFLAGS while VIF is in the noninterruptible state, the Pentium will respond to an STI instruction, by "automatically executing the STI instruction and awaiting an interrupt request by means of the emulation software without first changing the state of the VIF bit in EFLAGS from a Noninterruptible state to an Interruptible state." (UK Patent Application GB 2 259 794 A, Claim 1.c.)

As I've already mentioned, this information was publicly available at the time Intel was trying to keep it secret. Anybody with the necessary skills could use this information to reverse engineer the remaining details of Intel's VME implementation. In the long run, keeping this information secret hurt the engineering community more than it helped Intel maintain any competitive advantage.

VME Details Finally Released

When the Pentium Pro manuals were finally released in April 1996, Intel released its own description of VME. By this time, my article "Pentium's Virtual Mode Extensions Revealed" (coauthored with Jim Brooks, *EE Times*, November 11, 1995) describing VME had been publicly available for five months. Upon reading Intel's VME description, I noticed that some of it is fatally flawed. Consider the following excerpt from the *Pentium Pro Family Developer's Manual, Volume 3*, section 12.3.5:

> If the processor receives an interrupt or exception and the VIF flag is clear (maskable hardware interrupts enabled), the processor performs the same operation as it does for method 5 interrupt and exception handling (that is, it redirects handling to the 8080 program's interrupt and exception handlers). The processor also handles interrupts and exceptions in this manner if the VIF flag is set, and the processor receives either an NMI interrupt or an exception (interrupt vectors 0 through 18).
>
> If the processor receives a maskable hardware interrupt (interrupt vector 32 through 255) when the VIF flag is set, processor performs and the interrupt handler software must perform the following operations:

In some way or another, almost every sentence in this quoted text needs correction. The passage should read as follows:

> If the processor receives an INT-*n* instruction, the processor performs the same operation as it does for method 5 interrupt and exception handling (that is, it redirects handling to the 8086 program's interrupt and exception handlers). The processor never handles maskable hardware interrupts, exceptions, or an NMI interrupts in this manner.
>
> If the processor receives a maskable hardware interrupt, exception, or an NMI interrupt when the VIF flag is set, the processor causes a #GP(0). Otherwise, when the VIF flag is clear the processor and interrupt handler software must perform the following operations:

I was so confused by Intel's documentation that I'm still not sure what it was trying to say. Therefore I made my best guess, while making the technical corrections along the way. To determine if Intel has corrected these errors, I checked this same text in the newly released *Intel Architecture Software Developer's Manual, Volume 3*, discovering that Intel has completely removed this entire section of text, and the detailed explanation that followed.

The following text was removed from Intel's manuals. The text is the continuation of the passage just listed. The text describes the steps which the microprocessor and Ev86 monitor must perform when a maskable hardware interrupt, exception, or an NMI interrupt occurs when EFLAGS.VIF=0.

> 1. The processor makes a call to a protected mode interrupt handler as described in the following steps. These steps are almost identical to those described for method 1 interrupt and exception handling in "Handling a Virtual-8086 Mode Interrupt or Exception Through a Protected-Mode Trap or Interrupt Gate" on page 12-17:
>   a. Switches to 32-bit protected mode and privilege level 0.
>   b. Saves the state of the processor on the privilege-level 0 stack. The states of the EIP, CS, EFLAGS, ESP, SS, ES, DS, FS, and GS registers are saved (see Figure 12-5 on page 12-18). In the EFLAGS image on the stack, the IOPL field is set to 3 and the VIF flag is copied to the IF flag.
>   c. Clears the segment registers.
>   d. Clears the VM flag in the EFLAGS register.
>   e. Begins executing selected the protected-mode interrupt handler.
> 2. The recommended action of the protected-mode interrupt handler is to read the VM flag from the EFLAGS image on the stack. If this flag is set, the handler makes a call to the virtual-8086 monitor.
> 3. The virtual-8086 monitor reads the VIF flag in the EFLAGS register. If the flag is set, the virtual-8086 monitor sets the VIP flag in the EFLAGS register to indicate that there is an interrupt pending and returns to the protected mode handler.
> 4. The protected mode handler executes a return to virtual-8086 mode.
> 5. Upon returning to virtual-8086 mode, the processor continues execution of the 8086 program without handling the interrupt.
>
> When the 8086 program executes the STI instruction to clear the VIF flag, the processor does the following:
>
> 1. Checks the VIP flag.
>   a. If the VIP flag is clear, the processor clears the VIF flag.
>   b. If the VIP flag is set, the processor generates a general-protection exception (#GP).

2. The recommended action of the protected-mode general-protection exception handler is to then call the virtual-8086 monitor and let it handle the pending interrupt.

A typical action of the virtual-8086 monitor is to clear the VIF and VIP flags in the EFLAGS image on the stack and execute a return to the virtual-8086 mode (through the protected-mode exception handler). The next time the processor receives a maskable hardware interrupt, (providing the VIF flag is still clear) it will handle it in the same manner as with method 5 interrupt and exception handling.

Note that the states of the VIF and VIP flags are not modified in real-address mode or during transitions between real-address and protected modes.

More Information: Missing in Action

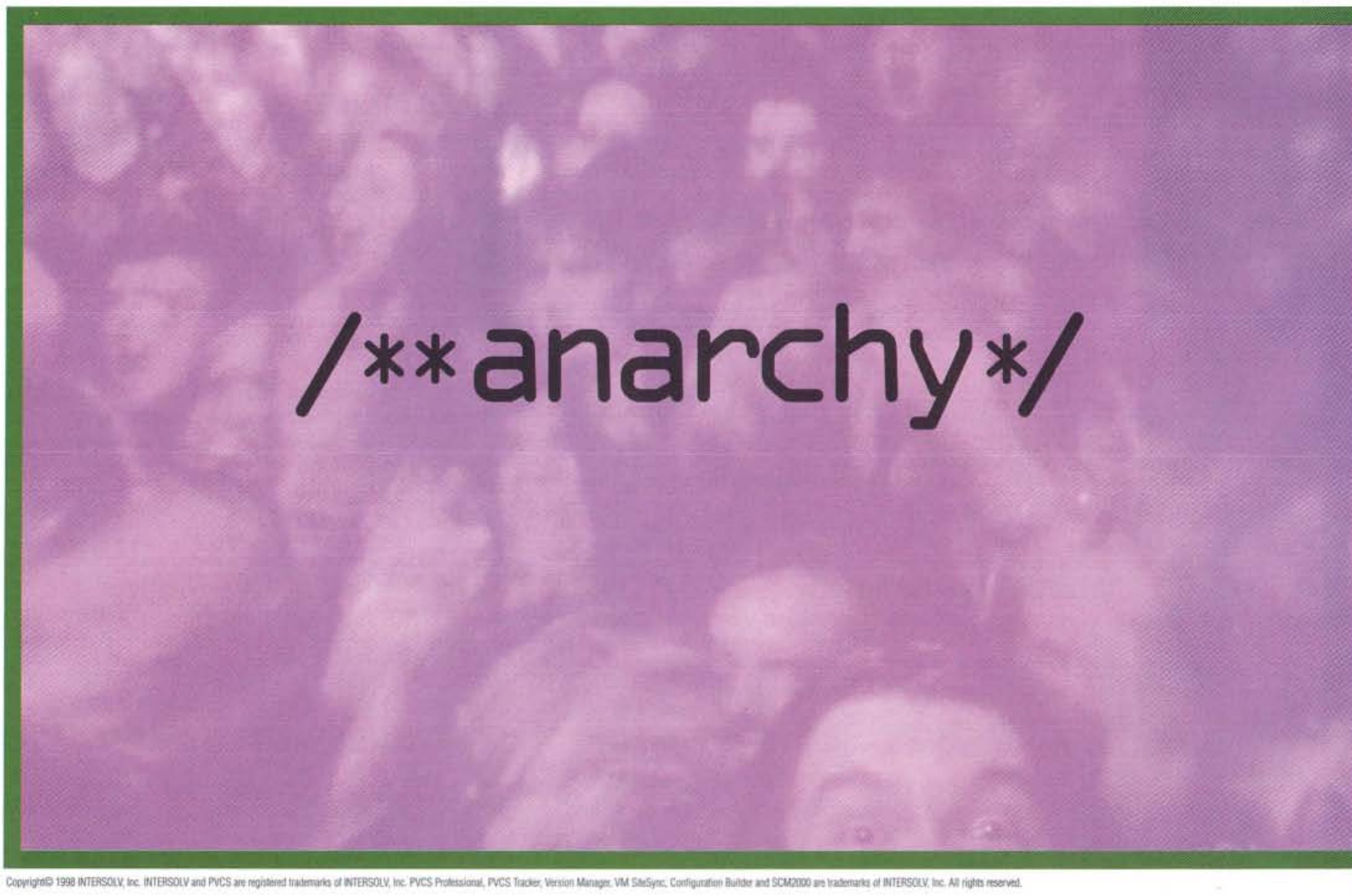
Intel has never released the algorithms used by the various CPU instructions needed to support VME. Perusal of Intel's Pentium manuals shows that CLI, STI, PUSHF, POPF, INT-n, and IRET have all been modified to support VME. This support is necessary, as all of these instructions have the ability to modify the Interrupt Flag in the EFLAGS register (EFLAGS.IF). With VME enabled, the processor must now act on the Virtual Interrupt Flag (EFLAGS.VIF), instead of the real IF flag.

To the best of my ability, I've made Listing One (listing begins on page 138) representative of the algorithms Intel uses in its Pentium VME implementation. I've only included the algorithms for CLI, STI, PUSHF, POPF, and IRET. I never attempted to reverse engineer the INT-*n* instruction. The INT-*n* instruction should closely resemble the PUSHF instruction, with the additional support needed for the Interrupt Redirection Bitmap. The code is written in pseudo-C format, but is intended to be human readable.

VME Caveats (When CR4.VME=1)

VME is wonderful for reducing the complexity of the operating system, but has many caveats. Some of the quirks are natural extensions of VME or the underlying architecture, while others make little sense.

- While an Ev86 task is running, the IF-sensitive flags commit fakery to fool the Ev86 task into thinking that it is actually setting and clearing the IF flag. When a fault occurs to the monitor, the monitor sees the actual values of VIF, VIP, and IF unfiltered. This behavior gives the monitor direct control over virtual interrupts, and hardware interrupts.
- Software-generated exceptions are not influenced by the IR bitmap. Instead, these opcodes will always invoke the protected mode exception handler. Software-generated exceptions are as follows:
  1. INT1, also known as ICEBP. (An undocumented opcode, 0xF1, which generates an INT-1 exception.)
  2. INT3 (opcode 0xCC).
  3. INTO (opcode 0xCE).
  4. BOUND (opcode 0x62).
- All processor exceptions invoke the protected mode exception handler. The IR bitmap never influences processor exceptions.
- Maskable hardware interrupts and NMI interrupts invoke the protected mode exception handler when EFLAGS.VIF=0. When EFLAGS.VIF=1, maskable hardware interrupts and NMI interrupts cause a #GP. The IR bit map never influences these interrupts.
- VIF will never transition from 0 to 1 while a virtual interrupt is pending when IOPL<3. This condition will always cause a general-protection fault (#GP(0)) before VIF is set. This can best be described as follows: While the Ev86 task is in an uninterruptible state (VIF=0), a timer tick occurs, and causes a transi-



Copyright© 1998 INTERSOLV, Inc. INTERSOLV and PVCS are registered trademarks of INTERSOLV, Inc. PVCS Professional, PVCS Tracker, Version Manager, VM SiteSync, Configuration Builder and SCM2000 are trademarks of INTERSOLV, Inc. All rights reserved.

tion to the protected mode interrupt handler. The timer-tick routine sets VIP on the stack frame, and returns to the Ev86 task. The Ev86 task performs an STI instruction, which will put the program into an interruptible state. Before VIF is actually set, the #GP occurs. (This behavior can be observed by inspecting the EFLAGS image on the CPL-0 stack frame.) This behavioral characteristic is the basis for Intel's British Patent application, which claims "if the VIP bit is in its 'pending' state, then the STI instruction can be executed and an awaiting interrupt is serviced by using the emulation software without first changing the VIF bit from its CLI state to its STI state."

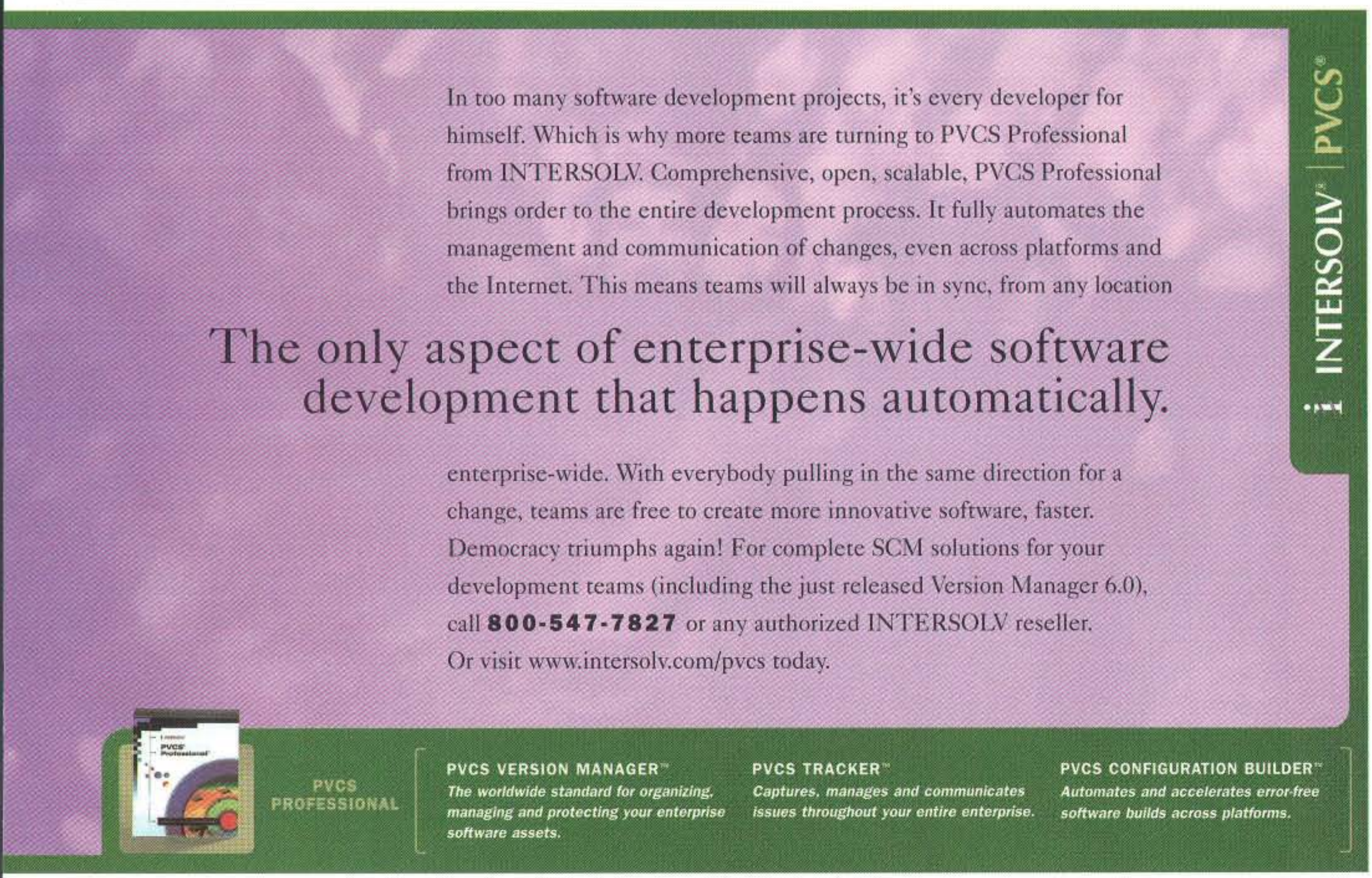
- VIP functions in IOPL=3 in a strange manner. When IOPL<3, and VIP=1, any transition from VIF=0 to VIF=1 will cause a general-protection fault before VIF is actually set. When IOPL=3, this behavior is different. VIF must actually be set before the #GP occurs.
- PUSHFD, POPFD, and IRETD will all generate a general-protection fault in an Ev86 task when IOPL<3.
- The behavior of POPF and IRET is inconsistent in how it handles the trap flag (TF) from the FLAGS image on the stack. When POPF is invoked with TF=1 on the stack image, a general-protection fault occurs. When IRET is invoked under the same condition, the general-protection fault does not occur. It has been recently reported that newer Pentiums do not demonstrate this behavior. If true, then Intel has unceremoniously fixed this behavior, and failed to mention this fact in their errata documents.
- IRET is no longer IOPL-sensitive when CR4.VME=1. Consider that the Ev86 task is running and an interrupt's associated IR bit is set. When this interrupt is invoked, it will cause a fault to the monitor, or the protected mode interrupt will be invoked (depending on IOPL). The interrupt is then reflected back to the Ev86 task. When IOPL=3, the behavior of the fault and subsequent IRET matches non-Ev86 behavior. When IOPL<3, the subsequent IRET from the interrupt service routine doesn't cause a fault back to the monitor, as IRET is no longer IOPL-sensitive. This behavior is inconsistent with its non-Ev86 counterpart.
- When IOPL=3, the CPU never updates VIF, though software can manipulate it in CPL-0. This can lead to the next caveat...
- The CPU will generate a general-protection fault (#GP(0)) whenever VIF=1, and VIP=1 and running in CPL-3. This is regardless of whether or not the code is an Ev86 task.
- In CPL-0, IRETD has the ability to set VIF and VIP, but POPFD does not.

Conclusion

I have described everything I know about VME in this series, but I'm sure that even so, not everything is known and documented. For example, since I wrote my last VME column, I discovered a few more caveats. As time goes on, I'm sure more VME details will be discovered. Yet in these three articles, I've presented many times more information than is contained in Intel's manuals. In the release of its newest architecture manual, Intel has demonstrated that it wishes to say less about VME, thereby removing vital information from the engineers who need it most. Therefore, consider these articles a blessing, and save the source code contained within. You never know when you'll need it—and you'll never know when Intel will release any more VME details.

DDJ

(Listing begins on page 138.)



AD LINK 227

Nimmerics

Dennis E. Shasha

I was just figuring out how I could force a draw in our chess game, when the doorbell rang. It rang again almost immediately. Then again.

"Must be the Feds," Ecco said with a sigh. "No patience at all."

Ecco's 10-year-old niece Liane, who had been watching our game with an ironic smirk, said "I'll get it," and ran towards the door.

Federal agent Thomas flashed his ID badge to all three of us, took a seat without being invited, opened his briefcase, pulled out a folder, and began to speak.

"We were keeping Wundermann under observation. Our agents saw him yesterday morning. By the afternoon, he was gone. All he left was this note. The trouble is that we don't know what to make of it."

Wim Wundermann was a professional colleague of Ecco's. He had dropped out of sight a few months earlier, saying that he had discovered patterns he called "Nimmerics." Ecco told him that he should steer clear of applications to encryption.

"Well, what does the note say?" Liane asked.

Dennis, a professor of computer science at New York University, is the author of The Puzzling Adventures of Dr. Ecco *(Dover, 1998),* Codes, Puzzles, and Conspiracy *(W.H. Freeman & Co., 1992),* Database Tuning: A Principled Approach *(Prentice Hall, 1992), and (coauthored with Cathy Lazere)* Out of Their Minds: The Lives and Discoveries of 15 Great Computer Scientists *(Springer Verlag, 1997). He can be contacted at DrEcco@ddj.com.*

Agent Thomas showed us the paper, and we all noticed that the writing was strange—especially at the beginning of the note.

> This is useful information for any Nim-Wise player. You know how to Play this Easy game of Nim Though it is not as easy as it seems.

"Oh, yes, I remember," Liane interrupted.

"You start with a bunch of stones and you can take one, two, or three of them in your turn. You win if you remove the last stone."

"Right," said Ecco, "and do you remember the winning strategy?"

Liane answered, "You try to end each turn leaving zero, four, eight, 12, 16,... (any multiple of four) stones to the other player. Then you force the number of stones to exactly four less at the end of your next turn."

"What would happen, Liane," Ecco asked, "if each player could remove one more stone than the last player? So, if the first player can remove up to three in the first move, then the second can remove up to four in the second move, and the first can remove up to five in the third move, then the second can remove up to six in the fourth move, and so on?"

"Hmmm," Liane paused for a moment. "The second player would still win if there were four stones at the beginning, but would also still win if there were five, since he'd be able to take up to four in the second move. I'll have to think about this some more."

"May I continue?" Federal agent Thomas demanded. "We are talking about the disappearance of a major national asset here and..."

"Yes, of course," Ecco apologized (though I thought I detected a trace of mockery in his voice). "Please go on."

Thomas read on, but in a louder, slower voice.

> I will give you two puzzles. Suppose I present a bowl containing one \$1 bill, two \$2 bills, three \$5 bills, and four \$10 bills. Play consists of moving a single bill at a time from the bowl to the table, which has no bills initially, until the amount on the table reaches or exceeds a given target number. If a player's move makes the amount reach that target, then that player wins. If a player's move makes the amount exceed that target, however, then that player loses.
>
> For example, if the target number is three, then the second player wins, because the first player will either exceed three dollars on his first move or will place a \$1 or \$2 bill on the table, in which case the second player will make the amount equal three dollars on her next move.

"Yes, but the first player usually has an advantage," Liane observed.

Thomas glared at her, but Ecco looked at her curiously.

"Oh, it just feels that way," she said with a shrug.

"It's true for Nim," she added.

Ecco nodded, but then turned his eyes to agent Thomas.

Thomas read on with even more deliberation.

Let X be the first target number 29 or greater in which the second player can force a win. Let Y be the first target number 41 or greater in which the first player can force a win. Find me at that location.

Can you find Wundermann's address at X and Y? Does either X or Y change if there are twice as many $1s, $2s, $5s, and $10s? If so, to what?

Ecco nodded his head thoughtfully, then said "Okay, so let's play Wundermann's game, Miss Feels-That-Way."

While Thomas paced the room, Ecco worked out the problem by playing the game with Liane. After he gained enough intuition, he gave Thomas the answer. "These two numbers are the X and Y you are to look for, perhaps they are street and avenue addresses. Liane has a theory about the letters."

"NWPENT," she said. "Look at the beginning of the note. Those are the weird uppercase letters. The others are there by grammar."

Federal agent Thomas stared at her as if puzzled for a moment, then his face cleared. He collected his papers, stood up, and left without a word.

A few days later, Thomas was back. "We found the place. A penthouse apartment in the Northwest district near the street corner you identified," Thomas said, nodding to Liane and Ecco. "Now, Wundermann has left a new puzzle. The guy's insane."

> Consider a variation of Nim which I'll call Expanding Nim. In that variation...

Thomas paused, then said, "This new game is exactly the one Ecco asked Liane about the other day. It is the standard Nim except that, for the first move, the first player can take one, two, or three stones; in the second move, the second player can take one, two, three, or four stones; in the third move, the first player can take one, two three, four, or five stones; in the fourth move, the second player can take one, two, three, four, five, or six stones, and so on. The possibilities expand at each move, making this a pretty difficult game to play."

The second player would still win if there were four stones at the beginning, but would also still win if there were five

"Well, Liane, can you do it?" Federal agent Thomas asked.

"Sure," she answered cheerfully. "Here they are..."

Can you find Liane's solution as well as the solution to the question posed earlier in italics? If so, let me know at DrEcco@ddj.com.

DDJ

Last Month's Solutions

Ecco believes in symmetry, so his solutions were to have the spacecraft land at the following coordinates:

For 7 spacecraft:	For 6 spacecraft:
(250 250.0	(250 250.0
250 500.0	250 500.0
250 750.0	250 750.0
500 500.0	750 250.0
750 250.0	750 500.0
750 500.0	750 750.0)
750 750.0)	

I don't think these are the best possible solutions, however. See if you can do better. The generalization to any number of points is an open problem so far.

A Whole Lotta *New* Stuff @ cdrom.com


Hand Held Computing Toolkit
Includes programs for hand held computers, such as the PalmPilot™, and Windows CE ... $39.95

Linux OS: The Professional Edition
Offers a ***complete*** solution for Linux users! Includes **14 CDROMs** and **2 books!** $149.95



Internet Mega-Pak
Contains ***everything*** you'll ever need to enhance your time on the World Wide Web. Incl. **6 CDROMs!** . . $69.95

SDE Courseware
Includes courseware modules based on common high-level programming themes $79.95


The Complete FreeBSD®, 2nd Ed.: This book contains 1,725 pages on installing & running FreeBSD. Includes **4 CDROMs**. . . . $49.95

X2FTP: The definitive games & graphics programming toolkit. $39.95

JOLT! Developer's Toolkit For Java™: Java-based language code translators, compilers/assemblers, script debuggers, tutorials, and more $39.95

3Dfx Mania: 3Dfx board compatible game add-ons, game patches, and more $29.95

UNIX Freeware: Programmer's source code to over 700 UNIX® packages $39.95

Linux Toolkit DVD-ROM: 6 CDROMs worth of data on one disc! You get over 31,500 Linux files in a total of 3.4 gigabytes $99.95

Walnut Creek CDROM
4041 Pike Lane Ste F-397
Concord, CA 94596

Toll Free: 1(800)786-9907
International: +1(925)674-0783
Fax: +1(925)674-0821
Email: orders@cdrom.com
Web: http://www.cdrom.com (397)

AD LINK 404

THE ELEVENTH INTERNATIONAL SOFTWARE QUALITY WEEK

Theme: Countdown To 2000

The Year-2000 question emphasizes how important it is to accomplish system changes and testing under pressure, against a deadline, and within real-world economic constraints.

Tutorials

Ten experts in the field share their insights.

A. *Metrics for Quality Assurance and Risk Assessment*
Dr. Linda Rosenberg & Mr. Larry Hyatt - Unisys/SATC GSFC NASA

B. *Current Techniques and Tools for Software Reliability Engineering*
Dr. Michael R. Lyu - The Chinese University of Hong Kong

C. *More Reliable, Faster, Cheaper Testing through Software Reliability Engineering*
Mr. John D. Musa - Consultant

D. *Formal Methods to Analyze and Improve Software Requirements Specifications*
Mr. Michael Deck - Cleanroom Software Engineering, Inc.

E. *An Overview of Testing*
Dr. Boris Beizer - Analysis, Inc.

F. *Software Metrics for Procedures, Objects, and Agents*
Mr. Hans-Ludwig Hausen - GMD German National Research Center for Information Technology

G. *Automating Software Testing and Reviews*
Mr. Ed Kit - Software Development Technologies

H. *Evolutionary Delivery Project Management*
Mr. Tom Gilb - Consultant

I. *Testing Strategies for Object-oriented Systems*
Mr. Robert V. Binder
RBSC Corporation

J. *Test Process Improvement*
Mr. Martin Pol - IQUIP Information B.V.

Keynote Speakers

Robert L. Glass,
Computing Trends

Cem Kaner,
Attorney at Law

Boris Beizer,
Analysis, Inc.

David Parnas,
Communications Research

Dave Duchesneau,
Boeing

Robert V. Binder,
RBSC Corporation

Program Chairman:
Edward Miller,
Software Research, Inc.

Who Should Attend

- Senior quality assurance managers looking for powerful maintenance and testing techniques.
- All quality assurance and testing specialists who need exposure to authoritative sources for software test technology.
- Maintenance technicians looking for techniques that control product degradation.
- Technologists who want to catch up on the state-of-the-art techniques in software testing, quality assurance, and quality control.

QW Presents

- **Technical Track**: Top Technical presentations about Test Automation, Criteria-Based, and Statistical Methods, Test Environments, and Java Compatibility Testing.
- **Applications Track**: Devoted to the applications of test methodologies: Web Testing, GUI Testing, Testware and Defects.
- **Tools & Solutions Track:** Listen to companies present their problems and real-world solutions.
- **Management Track**: Presentations aimed at managers of the SQ process, presented by their peers with real-life experiences from Bellcore, IBM, HP and more.
- **QuickStart Track:** 90-minute tutorials on questions, obstacles, ideas, and current practices in the industry.
- **Panel Sessions:** Co-Chaired by **Johanna Rothman,** Rothman Consulting **& Brian Lawrence,** Software Publishing
- **Birds-of-a-Feather:** Co-Chaired by **Danny Faught**, HP **& Brian Marick,** Testing Foundations

Tools Expo

Products and services that support software test methodology techniques will be displayed May 27 and 28.

- Exhibitors include: **Aonix**; Azor; Bellcore; Compuware; Dynamic Software Technology; ErgoLight; Hall Kinion; IEEE; Interworking labs; Keylabs; Performance Research; RSW; Software Development Technologies; **Software Research**; Softfront; Softbridge; Software Technology Labs; **Technology Builders** & Teradyne.

QW98
26-29 May 1998
San Francisco
California

SOFTWARE RESEARCH, INC.
Sponsor of Quality Week
since 1988

Check Website for FULL Program

WWW: http://www.soft.com
Email: qw@soft.com
Tel: (415) 957-1441
Fax: (415) 957-0730

Change Your View... Be A Lead Dog!

NuMega Technologies

Software Engineering

Are you ready to lead the pack? We are expanding the NuMega team to create the next wave of award-winning tools for Windows 95 and NT. If you're one of the best and need to be on the cutting edge of technology, consider one of the following opportunities.

Windows GUI Expert
proven experience in C/C++, MFC, and UI design and development methodologies is required. Knowledge of Windows internals a big plus.

Windows Internals expert
You'll need extensive knowledge of Intel x86 architecture, Windows internals.VxDs,and system-level code for this position. A proven system development track record in Assembler, C, and C++ required.

Software Programmers
Apprentice positions available for those with long-term goals of mastering their craft. Candidates should have strong communication skills and experience with PCs, Assembler, C, C++, Windows, and DOS

email your resume to jobs@numega.com, or send it to:
NuMega Technologies
9 Townsend West
Nashua, NH 03063
Att: Human Resources

Time for a new job?

Hi-Tech Jobs Fast www.dice.com

Don't gamble with your job search! Point your browser to **www.dice.com** for **FREE** access to thousands of contract and full-time job listings for Programmers, Analysts, Technical Writing professionals and more!

DATA PROCESSING INDEPENDENT CONSULTANT'S EXCHANGE

A service of D&L Online, Inc. 515.280.1144

MicroStamp™

BE A PART OF THIS EXCITING TECHNOLOGY

EMBEDDED SOFTWARE ENGINEER

Will develop applications, Arbitration Algorithms, and lower-level libraries for RF Identification communications on embedded interrogators. Qualified candidates will have a BSCE, BSCS, BSEE degree or greater and 5+ years professional C language experience. Experience in embedded processor programming, communications programming, basic digital design, and DSP algorithm programming preferred.

APPLICATION SOFTWARE ENGINEER

Will develop RF identification applications using lower-level RIC libraries. This engineer will develop and maintain applications using C and C++ for MS Windows. Qualified candidate will have BSCE, BSCS, BSEE or greater and 5+ years of professional C and C++ language experience. Experience in MS Windows, communications programming, basic digital design, and programming with MS MFC preferred.

MICRON COMMUNICATIONS,INC.

**960 Broadway Ave, Suite 500
P.O. Box 6, MS942-BC
Boise, ID 83707-0006
Call: (208) 333-7324
Fax: (208) 333-7466
www.microncommunications.com
EEO/AA**

Global Resource Management

Top Dollar Paid For Technical Talent

C++, UNIX, Solaris, Java, Web, Win NT

$110,000

Don't miss our referral bonus. If you refer a collegue and we hire the candidate, you will receive a **$4000** commission.
Email: Resumes@resourcemgmt.com
(800)408-8005 (voice) (888)408-8006 (fax)

Since 1979 Spectrum Concepts has enabled its clients to effectively meet their business objectives by providing leading-edge information technology consulting expertise. We have opportunities for talented professionals with the following skills:

- C++, OOD/OOP
- Java, Smalltalk
- Corba, Orbix
- Visual C++/MFC, Win32
- ActiveX, COM/DCOM
- Visual Basic, PowerBuilder

For NY: (212)791-4800 Fax: (212)791-6639
recruitingNY@sconcepts.com

For NJ: (732)603-0900 Fax: (732)603-0110
recruitingNJ@sconcepts.com

For PA: (215)643-6000 Fax: (215)643-8600
recruitingPA@sconcepts.com

www.sconcepts.com

Designers
Developers
Architects

SPECTRUM concepts CONSULTING CORP

SPECIALIZING IN HIGH-TECH SEARCH AND RECRUITING FOR 14 YEARS

RICCIONE & ASSOCIATES, INC.

972.380.6432 (V)
972.407.0659 (F)

hitec@riccione.com

Full-Time and Contract Positions:

- Win NT/95
- UNIX
- Real-Time
- Applications
- Systems-Level
- Firmware
- SQA/Test
- Product Mgmt.
- SW Mgmt.
- Telecom
- Datacom
- Vendor SW
- Internet
- Multimedia
- Control Systems

VISIT OUR WEB SITE

RICCIONE ON-LINE COMPUSEARCH

WWW.RICCIONE.COM

Sexy

Job lost its sex appeal?
Bored? Need a new challenge?
Want more dough?
Ready to tell your manager @#$!
Do you really need the headache?

Let us help. At Peak Career Mgmt., we specialize in recruiting software developers (C/C++, Windows, UNIX, Device Drivers) and other software industry professionals across the US.

Our services are free to applicants. Positions available nationwide.

Contact us today at
dobbs@peakweb.com
(303)316-0800, FAX (303)316-0700

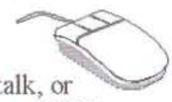
www.peakcareer.com/peakcareer/

THE SHAY GROUP

Visual C++ Delphi Powerbuilder Smalltalk Oracle

. . . GUI, GUI, GONE

If you're a software developer skilled in designing graphical user interfaces with:



- Visual C++,
- Powerbuilder,
- Delphi, Smalltalk, or
- Oracle Developer 2000

. . . and you're ready to make a change in your employment picture-- double click here--your window of opportunity.

THE SHAY GROUP
voice: 800/894-2888
fax: 413/256-8582
75114.2251@compuserve.com
http://www.shaysnet.com/~shayg/

http://www.scientific.com

Professional Software Developers
Looking for career opportunities?
Check out our website!
Nationwide Service
Employment Assistance
Resume Help
Marketability Assessment
Never a fee

Scientific Placement, Inc.

800-231-5920 800-757-9003 (Fax)
das@scientific.com

S/W ENGINEERING POSITIONS NATIONWIDE

We Understand The Programmer's Mind.℠

When the country's top firms look for the best developers available, they turn to Bateman. Why? Because we specialize in MS Windows, NT, JAVA and Macintosh recruiting nationwide. So if it's time for a career move, give us a call. We understand your skills, and the marketplace for them... we understand *you.*

Bateman Inc.
711 Daily Dr., Suite 106
Camarillo, CA 93010
Tel: 805-383-3338 Fax: 805-383-3337

Great Jobs in Lake Tahoe!

Zephyr Associates develops the leading investment analysis software for the institutional investment market. Located in one of the most beautiful places in the world, we are minutes from skiing, hiking, and boating.

We are looking for experienced Senior C++ application developers to join team embarking on next generation of product. Proficiency developing Windows applications using MFC is required. Experience in object oriented design and development desired.

We offer an attractive salary, benefits, and an outstanding retirement plan.

Zephyr Associates, Inc.
E-mail: aaron@styleadvisor.com
www.styleadvisor.com/jobs.html

1000's of High Tech Jobs in your area @ *YOUR* address for SUCCESS:

infoworks
The Computer Job Center

Job Seekers

Job Search — Search for a job that best matches YOUR skills and experience
Submit/Update a Resume — Point click EXPOSURE
Featured Companies — Check out the companies that use our services
Skills Quiz — Challenge Yourself!

Submit a resume on-line

Employers

infoworks Information *Member Entrance* *Take a Test* *Contact Us*

www.IT123.com

"The only regionally-specific technical career site on the web at no cost to you!"

Fax: 1-888-653-8145 E mail: info@Infoworksusa.com

RSVP SERVICES

Employment Service

trusted by computer professionals and employers since 1966

Clients and /or affiliates in all regions of U.S. and Canada • All platforms and technologies • Employment fees and relocation assistance paid by employing organizations

APPLY ONLINE:
www.rsvpjobs.com

or call (toll free), mail, fax or email resume: Howard Levin, Dept DDJ, RSVP Services, PO Box 8369, Cherry Hill NJ 08002-0369
Phone: 800/222-0153
Fax : 609/667-2606
Email: hl@rsvpjobs.com

Software Careers Ad Links

Ameritech	Ad Link 700
Anderson Consulting	Ad Link 701
IBASE Consulting	Ad Link 702
Bateman, Inc.	Ad Link 703
The Computer Jobs Store	Ad Link 704
D & L Online	Ad Link 705
Global Resource Management	Ad Link 706
Hamilton Technical Personnel	Ad Link 707
Infoworks USA	Ad Link 708
Micron Communications, Inc.	Ad Link 709
NuMega Technologies	Ad Link 710
Peak Career Management	Ad Link 711
Qualcomm	Ad Link 712
Riccione & Associates, Inc.	Ad Link 713
RSVP Services	Ad Link 714
Scientific Placement, Inc.	Ad Link 715
The Shay Group	Ad Link 716
Spectrum Concepts Consulting	Ad Link 717
User Technology Assoc., Inc.	Ad Link 718
Zephyr Associates	Ad Link 719

DDX/DDV

Listing One

```
//{{AFX_DATA(ECCircle)
enum { IDD = IDD_CIRCLE };
CString m_Px;
CString m_Py;
CString m_Radius;
//}}AFX_DATA
```

Listing Two

```
//{{AFX_DATA(ECCircle)
enum { IDD = IDD_CIRCLE };
CMacro  m_Px;
CMacro  m_Py;
CMacro  m_Radius;
//}}AFX_DATA
```

Listing Three

```
void ECCircle::Serialize(CArchive& ar)
{
    if (ar.IsStoring())
   {  // storing code
        ar  <<  m_Px
            <<  m_Py
            <<  m_Radius;
    }
    else
    {  // loading code
        ar  >>  m_Px
            >>  m_Py
            >>  m_Radius;
    }
}
```

Listing Four

```
CMFElem& ECCircle::operator=(const CMFElem& elRHS)
{
    CMFElem::operator=(elRHS);  // Call base class assignment 1st.
    if (this == &elRHS)          //Nothing to do !
        return *this;
    m_Px  =  ((ECCircle&) elRHS).m_Px;
    m_Py  =  ((ECCircle&) elRHS).m_Py;
    m_Radius  =  ((ECCircle&) elRHS).m_Radius;
    return *this;
}
```

C PROGRAMMING

Listing One

```
// ------ playmidi.cpp
#include <fstream>
#include <iostream>
#include "midifile.h"

class WmTellOverture : public MIDIFile {
    void StartTrack(int trackno);
    static int m_nNotes[][2];
public:
    WmTellOverture(std::ofstream& rfile) :
                MIDIFile(rfile, 1, 3, 120) { }
};
static const int iv = 100;  // note interval
static const int nt = 60;   // 1st note of song
int WmTellOverture::m_nNotes[][2] = {           // {note, delta}
    {nt,0},{nt, iv/2},{nt,iv/2},{nt,iv},{nt,iv/2},{nt,iv/2},
    {nt,iv},{nt, iv/2},{nt+5,iv/2},{nt+7,iv},{nt+9,iv},{nt,iv},
    {nt,iv/2},{nt,iv/2},{nt,iv},{nt,iv/2},{nt+5,iv/2},{nt+9,iv},
    {nt+9,iv/2},{nt+7,iv/2},{nt+4,iv},{nt,iv},{nt,iv},{nt, iv/2},
    {nt,iv/2},{nt,iv},{nt, iv/2},{nt,iv/2},{nt,iv},{nt,iv/2},
    {nt+5,iv/2},{nt+7,iv},{nt+9,iv},{nt+5,iv},{nt+9,iv/2},
    {nt+12,iv/2},{nt+10,iv*3},{nt+9,iv/2},{nt+7,iv/2},
    {nt+5,iv/2},{nt+9,iv},{nt+5,iv},
    {0,iv}
};
void WmTellOverture::StartTrack(int trackno)
{
    switch (trackno)    {
        case 1:
            // ---- tempo track
            TextEvent(0, META_SEQTRKNAME, 21, "William Tell Overture");
            Tempo(0, 250000);
            break;
        case 2:
            // ---- piano track
            TextEvent(0, META_SEQTRKNAME, 5, "Piano");
            ProgramChange(0, 0, 0);      // channel 0 = acoustic piano
            // --- play the notes
            int i;
            for (i = 0; m_nNotes[i][0] != 0; i++)    {
                if (i > 0)
                    NoteOff(m_nNotes[i][1],0,m_nNotes[i-1][0],0);
                NoteOn(0,0,m_nNotes[i][0],64);
            }
            NoteOff(m_nNotes[i][1],0,m_nNotes[i-1][0],0);
            break;
        case 3:
        {
            // ---- drum track
            const int shot = 42;
            const int crash = 49;
            TextEvent(0, META_SEQTRKNAME, 5, "Drums");
            ProgramChange(0, 9, 0);
            for (i = 0; m_nNotes[i][0] != 0; i++)
                NoteOn(m_nNotes[i][1],9,shot,64);
            NoteOn(iv,9,crash,100);
            break;
        }
        default:
            break;
    }
}
int main()
{
    std::ofstream ifile("WTO.mid", std::ios::binary);
    WmTellOverture wto(ifile);
    wto.WriteMIDIFile();
    return 0;
}
```

ALGORITHM ALLEY

Listing One

```
// Singly Linked List Quicksort
private Node QuickSort(Node before, Node first, int n) {
  int Num1=0, Num2=n, i=1;
  Node Pivot=first, aNode=first, aNodePrev=first;
  // Pivot Advancement
  for (i=1; i<n; i++, aNode=aNode.next) {
    if (aNode.compareTo(aNode.next) > 0)
      break;
    if ((i&1)==0) {
      Pivot = Pivot.next;
      Num1++;
    }
  }
  // Recognize sortedness in linear time
  if (i == n) return first;
  // Partition list by unlinking nodes with values less
  // than the pivot and pushing them onto front of list
  for (aNodePrev = aNode; i < n; i++) {
    aNode = aNodePrev.next;
    if (Pivot.compareTo(aNode) > 0) {
      aNodePrev.next = aNode.next;
      aNode.next = first;
      first = aNode;
      Num1++;
    }
    else aNodePrev = aNode;
  }
  if (before!=null) before.next = first;
  Num2 = n - Num1 - 1;
  // Recurse to sort sublists
  if (Num1 > 1) first = QuickSort(before, first, Num1);
  if (Num2 > 1) QuickSort(Pivot, Pivot.next, Num2);
  return first;
}
```

Listing Two

```
// Singly Linked List Recursive Merge Sort
private void mergesort(Node before, Node F1, int N1, NodePair NP) {
  if (N1 <= 1)
    NP.first = NP.last = F1;
  else {
    Node F2;
    int N2;
    N2 = N1; N1 >>= 1; N2 -= N1;
    mergesort(before, F1, N1, NP);
    F1 = NP.first;
    F2 = NP.last.next;
    mergesort(NP.last, F2, N2, NP);
    F2 = NP.first;
    merge(before, F1, N1, F2, N2, NP);
  }
}
//==============================================================
private void merge(Node before, Node F1, int N1,
                   Node F2, int N2, NodePair NP) {
  Node first=null, last=null, temp=null;
  int I, J;
  first = last = F1.compareTo(F2) <= 0 ? F1 : F2;
  for (I=J=0; I < N1 || J < N2; ) {
    if (I < N1 && (J>=N2 || F1.compareTo(F2) <= 0))
        { temp = F1; F1 = F1.next; I++; }
    else { temp = F2; F2 = F2.next; J++; }

    last.next = temp;
    last = temp;
  }
  if (before == null) First = first;
  else before.next = first;
  last.next = F2;
  NP.first = first;
  NP.last = last;
}
```

Listing Three

```
// Simpler Non-recursive Merge Sort
// NOTE: Use same merge() as in Listing Two
private void nr2_mergesort() {
  int i, j, k, N1, N2;
  Node F1, F2, before;
  NodePair NP = new NodePair();
  for (i=1; i < NumNodes; i<<=1)
    for (before=null, N1=N2=i, j=0; j+N1<NumNodes; j+=i<<1) {
      F1 = F2 = before==null ? First : before.next;
      for (k=0; k<N1; k++)
           F2 = F2.next;
      if (N2 > NumNodes-j-N1)
          N2 = NumNodes-j-N1;
      merge(before, F1, N1, F2, N2, NP);
      before = NP.last;
    }
}
```

Listing Four

```
// Singly Linked List Binary Search
public Node binarySearch(Object SearchKey) {
  Node PartitionFirst=First, MidPtr=null;
  int  PartitionSize=NumNodes, Mid, I, Result;
```

(continued on page 138)

(continued from page 137)

```
  while (PartitionSize > Ø) {
    Mid = PartitionSize / 2;
    MidPtr = PartitionFirst;
    for (I=Ø; I < Mid; I++)
      MidPtr = MidPtr.next;
    Result = MidPtr.compareTo(SearchKey);
    if (Result > Ø)
      PartitionSize = Mid;
    else if (Result < Ø) {
      PartitionSize -= Mid;
      PartitionFirst = MidPtr;
    }
    else return MidPtr;
  }
  return null;
}
```

UNDOCUMENTED CORNER

Listing One

```
CLI {                        /* v86 mode                            */
    if (EFLAGS.IOPL == 3) {
    then EFLAGS.IF = Ø;
    else if (CR4.VME == Ø) {    /* IOPL < 3                         */
        then #GP(Ø);            /* ERROR CODE = Ø                   */
        else EFLAGS.VIF = Ø;
        }
    }
}
STI {                        /* v86 mode                            */
    if (EFLAGS.IOPL == 3) {
    then EFLAGS.IF = 1;
    else if (CR4.VME == Ø) {    /* IOPL < 3                         */
        #GP(Ø);              /* ERROR CODE = Ø                      */
            else {
        if (EFLAGS.VIP == 1) {  /* if VIP already=1, #GP(?)         */
            then #GP(Ø);    /* ERROR CODE = Ø                       */
                    /* Processor never set VIF                  */
                    /*   before #GP()                           */
            else EFLAGS.VIF = 1;
                }
            }
        }
    }
}
PUSHF {                      /* v86 mode                            */
    if (EFLAGS.IOPL == 3) {
    then
    if (OperandSize == 32) {
        then push(EFLAGS & ØxFCFFFF);   /* Clear VM & RF            */
        else push(FLAGS);
    }
                    /* IOPL < 3                             */
    else if ((CR4.VME == Ø) || (OperandSize == 32)) { /* IOPL < 3  */
        then #GP(Ø);            /* ERROR CODE = Ø                   */
        else {
                TEMP = FLAGS;
        TEMP = TEMP OR Øx3ØØØ;  /* Set IOPL=3 on stack (dumb?) */
        TEMP.IF = EFLAGS.VIF;
        push(TEMP);
            }
        }
    }
}
POPF {                       /* v86 mode                            */
    if (EFLAGS.IOPL == 3) {     /* IOPL = 3                         */
    then if (OperandSize == 32) {
        then {
       TEMP = pop();
        /* Clear these fields from EFLAGS stack image:              */
        TEMP = TEMP AND NOT Øx1BBØ2A;   /* VIP VIF VM RF IOPL       */
        /* Keep these fields from current EFLAGS register:          */
        EFLAGS = EFLAGS AND Øx1B3ØØ2;   /* VIP VIF VM RF IOPL       */
        EFLAGS = EFLAGS OR TEMP;
        }
        else {
        TEMP = pop();
        /* Clear these fields from EFLAGS stack image:              */
        TEMP = TEMP AND NOT ØxBØ2A; /* IOPL                         */
        /* Keep these fields from current EFLAGS register:          */
        FLAGS = FLAGS AND Øx3ØØ2;   /* IOPL                         */
        FLAGS = FLAGS OR TEMP;
        }
    }
    else if ((CR4.VME == Ø) || (OperandSize == 32)) { /* IOPL < 3    */
            then #GP(Ø);
        else if ([SP].TF) {     /* If stack image TF=1, then #GP     */
                then #GP(Ø);
        else if ((VIP == 1) && [SP.IF]) {
                    then #GP(Ø);
                    else {
                        TEMP = pop();
            EFLAGS.VIF = TEMP.IF;
            /* Clear these fields from EFLAGS stack image:              */
            TEMP = AND NOT ØxB22A;  /* IOPL, IF                         */
            /* Keep these fields from current EFLAGS reg                */
            FLAGS = FLAGS AND Øx32Ø2;/* IOPL, IF                        */
            FLAGS = FLAGS OR TEMP;
                    }
                }
            }
        }
    }
}
IRET {                                  /* v86 mode                         */
    if (EFLAGS.IOPL == 3)               /* IOPL = 3                         */
        then if (OperandSize == 32) {
            then {
                EIP = pop();
                CS = pop();
                TEMP = pop();
                /* Clear these fields from EFLAGS stack image:              */
                TEMP = TEMP AND NOT Øx1BBØ2A;   /* VIP VIF VM RF IOPL       */
                /* Keep these fields from current EFLAGS register:          */
                EFLAGS = EFLAGS AND Øx1B3ØØ2;   /* VIP VIF VM RF IOPL       */
        EFLAGS = EFLAGS OR TEMP;
            }
            else {
                IP = pop();
                CS = pop();
                TEMP = pop();
                /* Clear these fields from EFLAGS stack image:              */
                TEMP = TEMP AND NOT ØxBØ2A;     /* IOPL                     */
                /* Keep these fields from current EFLAGS register:          */
                FLAGS = FLAGS AND Øx3ØØ2;       /* IOPL                     */
                FLAGS = FLAGS OR TEMP;
        }
        }
        else if ((CR4.VME == Ø) || (OperandSize == 32)) { /* IOPL < 3      */
            then #GP(Ø);
            else if ([SP].TF) {         /* If stack image TF=1, then #GP  */
                then #GP(Ø);
        else if ((VIP == 1) && [SP.IF]) {
            then #GP(Ø);
            else {
            IP = pop();
            CS = pop();
            TEMP = pop();
            EFLAGS.VIF = TEMP.IF;
            /* Clear these fields from EFLAGS stack image:      */
            TEMP = AND NOT ØxB22A;  /* IOPL, IF                 */
            /***************************************************/
            /* ** NOTE ** The following treatment of TF flag   */
            /* ********** *MAY BE A BUG* in the Pentium.       */
            /* POPF uses the FLAGS mask of Øx32Ø2, but IRET    */
            /* uses Øx33Ø2.  Consider that POPF and IRET       */
            /* #GP on a condition involving TF.  So is it      */
            /* just coincidence that IRET has further          */
            /* special treatment of TF on IRET, where POPF     */
            /* does not?                                       */
            /***************************************************/
            /* Keep these fields from current EFLAGS reg       */
            FLAGS = FLAGS AND Øx33Ø2;/* IOPL, IF, TF            */
            FLAGS = FLAGS OR TEMP;
            }
          }
        }
      }
    }
}
```

DDJ

AD LINK 309

PC EXPO in New York.

Where the corporate world comes to shop.

If you're a corporate professional looking for real solutions to your most challenging IT problems, PC EXPO is the one event that delivers all the goods.

More than 800 IT suppliers will come to PC EXPO with one thing in common: **corporate IT solutions – and to attend over 50 sessions within our Corporate Education Program.**

We deliver virtually every technology for serious profit - along with the resources and information you need to leverage them. From the desktop to the enterprise, it's all here. Hardware. Software. Peripherals. Notebooks. Servers. Storage. Networking. Telecommunications. Mobile/wireless. Computer telephony. NT. Java.

We mean all. Including an explosion of new Web and e-commerce solutions at our best-ever WEB.X – The Internet Event for Business, held concurrently during PC EXPO.

The world's full of IT events. But for corporate professionals, there's really only one. PC EXPO in New York.

To attend, call 800-829-3976, ext. 2980 or register online at www.pcexpo.com.

PC EXPO®

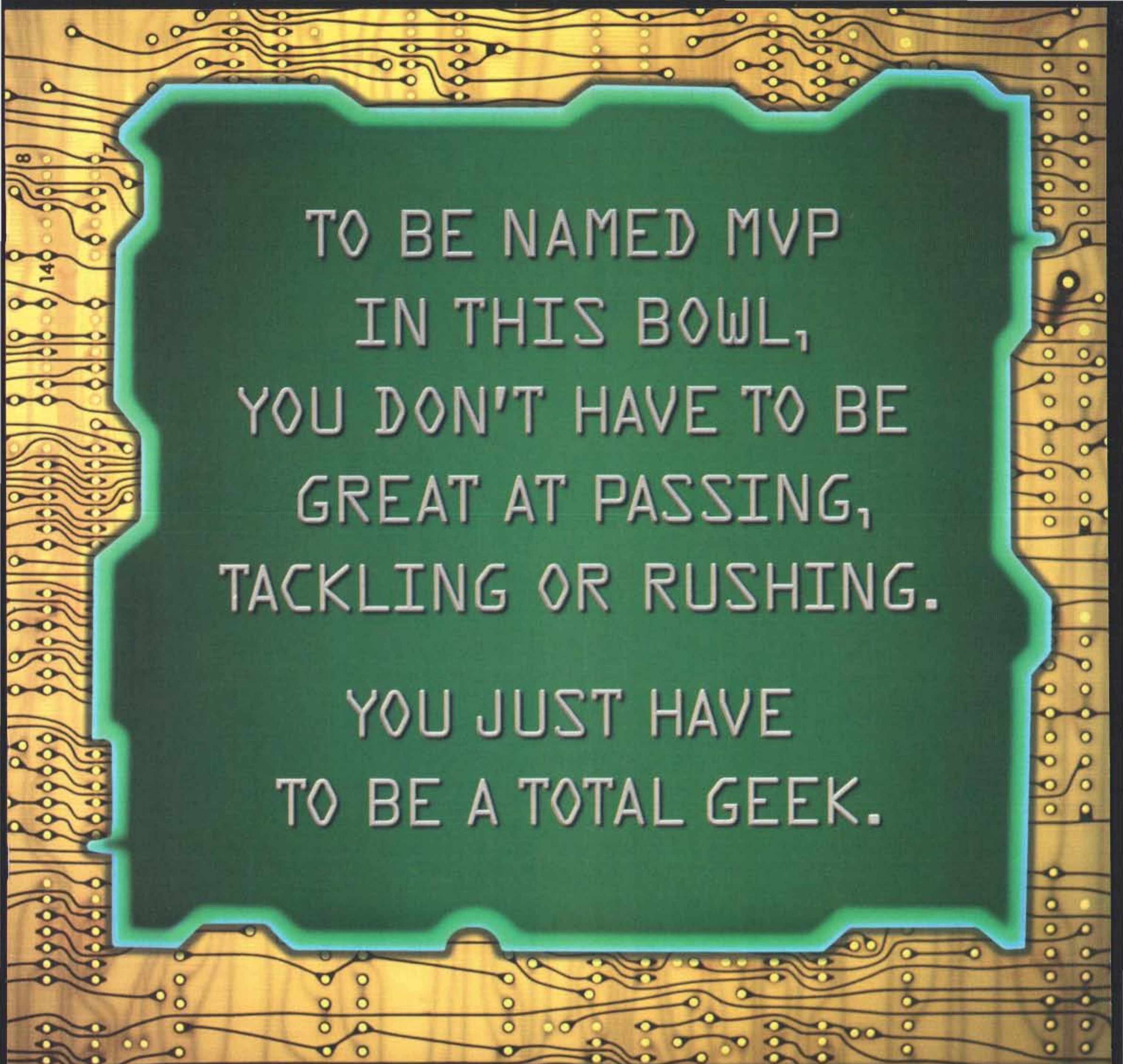
Exhibitors! Reach one-half-*trillion* dollars in IT buying power!

PC EXPO Exhibits • June 16-18, 1998 • Jacob K. Javits Convention Center • New York City
Corporate Education/Conference Program • June 15-18, 1998 • Held at the Marriott Marquis

Miller Freeman PC EXPO in New York is a registered trademark of Miller Freeman, Inc. • One Penn Plaza • New York, NY 10119

PC EXPO and WEB.X are registered trademarks of Miller Freeman, Inc. • One Penn Plaza – 11th Floor • New York, NY 10119 • 800-829-3976 ext. 2980 • 212-714-1300 • Fax 212-643-4802
PC EXPO is for trade professionals only. No one under 18 admitted. Your badge and carrier are non-transferable and will be confiscated and/or terminated upon any attempt to transfer or sell them.

Source Code: MK98



ZD ZIFF-DAVIS *presents*

THE ? COMPUTER BOWL®

It's The Tenth Annual Computer Bowl, where computer industry leaders from the east and west coasts battle for the title of "Super Sages of Cyberspace." They compete to earn bragging rights, but more importantly, to benefit the education and preservation efforts of The Computer Museum. For tickets, broadcast dates and complete information, or to just brush up on your trivia, check out our website at www.computerbowl.org.

Underwritten by: Bay Networks *Where Information Flows.* intel. The Computer Inside. Created by: the computer museum

Official sponsors: Arnold Communications boston.com Coopers &Lybrand The Cyber Angel EXR Lizard Lounge Graphics, Inc. Fidelity Investments Forrester infoseek Kleiner Perkins Caufield & Byers

COMPAQ Fleet Ready When You Are Lotus. acm NYSE Print Lexmark Secure Computing Nobody Comes Close Stratus The Availability Company Thomson Electronic Settlements Group Motorola US Web Yoyodyne

Fuzzy Logic

Peter N. Roth

Fuzzy Engineering

Bart Kosko
Prentice-Hall, 1997
547 pp., $80.00
ISBN 0-131-24991-6

Applications of Fuzzy Logic: Towards High Machine Intelligence Quotient Systems

Mohammad Jamshidi, Andri Titli, Lotfi Zadeh, Serge Boverie (editors)
Prentice-Hall, 1997
423 pp., $74.00
ISBN 0-133-62831-0

The Design and Development of Fuzzy Logic

Byron Miller
Impatiens Publications, 1997
142 pp., $52.00
ISBN 0-963-66370-4

Computationally, it is convenient to represent logical values with a single bit—False being represented by 0, and True by 1. Fuzzy logic does not refute reality, but rather extends the convenient binary representation into the floating-point range [0.0...1.0]. Thus, it can be said that perception becomes part of the computational milieu. "The sky is blue" may be 95 percent true at 10 AM, and five percent true at sunset. The paradox point is 0.5; at that point, truth and falsity are equal.

When designing a system to control a washing machine, for instance, you use perception to state what the system is to do. At the early stages of design, these statements take the following form: If the wash water is dirty, then add more detergent. This *if-then* statement is called a rule in a fuzzy system. It's fuzzy, because quantities are not mentioned (yet). Usually, several overlapping rules are used to model an entire system, so a second rule controlling the cleaning system might be as follows: If the wash water is very dirty, then add a lot more detergent. The difficulty, and art, of control-system design is determining the meaning of "water is dirty," and determining how much detergent is "more detergent." One might think of a fuzzy cleaning system as one that takes a poll of water quality and detergent quantity experts, and acts on some combination of their advice.

Because these are computerized systems, the programs can revise their own rules. Thus, fuzzy systems can be said to learn how to improve their own responses. The study of fuzzy logic is an attempt to codify these ideas, and place the discipline on a firm theoretical basis such that control systems can be built economically and confidently.

I first heard of fuzzy logic in 1993, when a friend urged me to read the semipopular treatment of the topic in *Fuzzy Logic: The Revolutionary Computer Technology That Is Changing Our World*, by Daniel McNeill and Paul Freiberger (Simon & Schuster, 1994). Given the pace of change these days, it seemed time to revisit the subject.

Fuzzy Engineering

Fuzzy Engineering, by Bart Kosko, is organized into 15 chapters. Each chapter presents a single topic at what I would guess is a graduate level (this comes as no surprise, considering that Kosko is a professor at the University of Southern California). The mathematics in all the chapters is challenging.

A list of references follows each chapter. The overwhelming majority of the references date from 1985, and the most-recent reference in each chapter is (at most) two years old, so the material is current. In addition to Kosko, there are 11 contributors to the text. While the book claims a single editorial control, there is some overlap and repetition of material. Each chapter repeats the mantra "a fuzzy system is a set of fuzzy *if-then* rules that maps inputs (the *if* parts) to outputs (the *then* parts)."

Each chapter concludes with homework problems. I would have appreciated a section that gave answers to, say, odd-numbered problems. That there is no such section suggests that the book was not designed primarily for self study, but to be used in a fuzzy-logic curriculum.

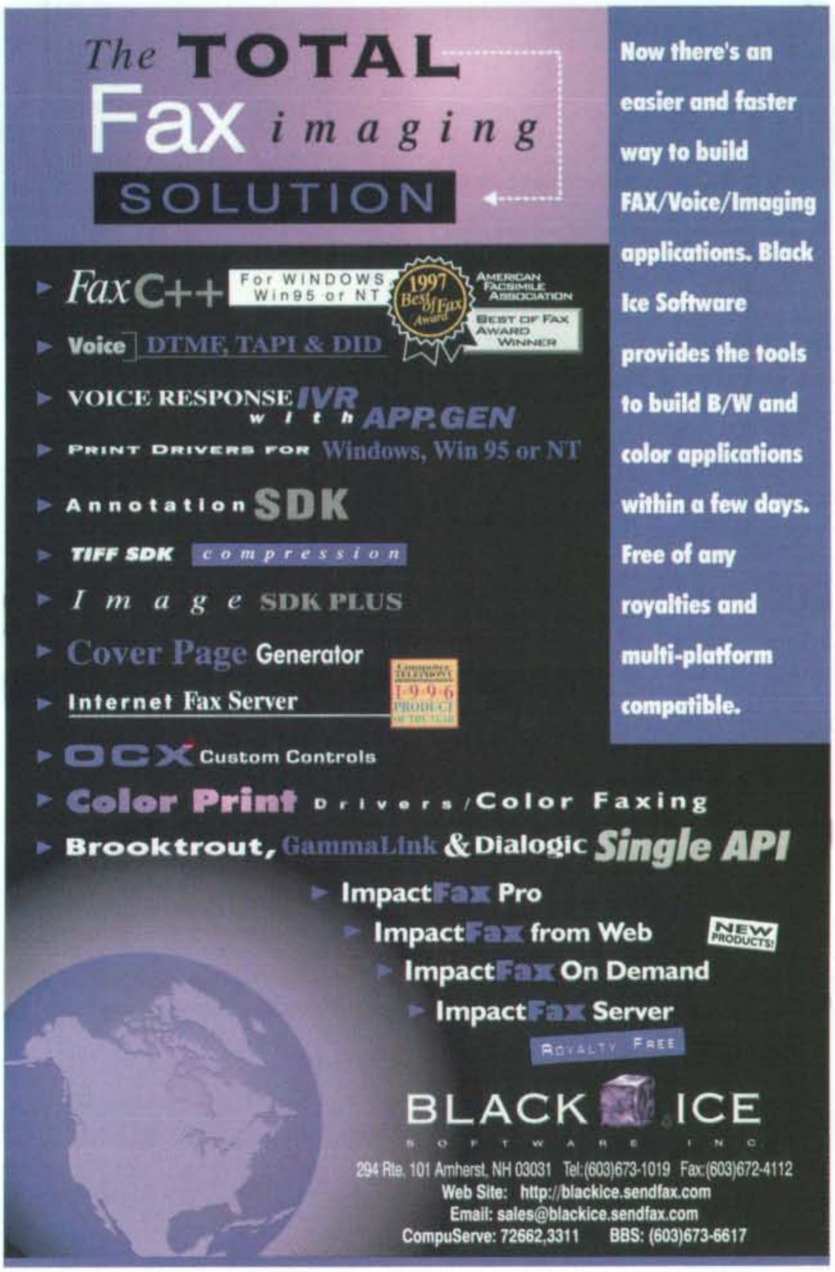
Kosko's focus is the Standard Additive Method (SAM). In this fuzzy-logic technique, the *then* parts of the rules are simply summed, rather than being combined with more elaborate techniques. Although this simplifies the computations, the SAM approach is not without its difficulties. Because good approximations require many rules, the SAM can suffer from rule explosion—that is, it may require more rules to approximate a good solution than can be computed in minimal time or with the computing facilities allocated to the job. Attacks on this problem are made in several ways: by altering the shape of the rules such that they cover ellipsoidal patch-

Peter is president of Engineering Objects International, producers of commercial and custom Delphi and C++ components. He can be contacted at http://www.inconresearch.com/eoi/ or peteroth@erols.com.

es rather than rectangles; by training the systems to learn and adapt their solutions; by making some parts of the system "smarter"; and so on. Programmers will recognize these strategies as attempts to deal with the law of the conservation of hair, which is stated roughly: In a hairy computation, you can move the hair around, but the quantity of hair remains constant.

Other chapters develop and apply fuzzy-logic theory to chaos, recursive partitioning, signal processing, communication, hardware, and cognitive maps. Chapter 12, entitled "Fuzzy Cubes And Fuzzy Mutual Entropy," is a tour de force. Given that a single fuzzy variable can be mapped continuously over the [(0,0),(1,0)] line, then two fuzzy variables map to the unit square [(0,0),(1,0),(1,1),(0,1)]; three map to the unit cube; and n variables map to the n-dimensional hypercube. The information state within the fuzzy cube is continuous, with the center paradox point being the fuzziest of the fuzzy. Kosko deduces from the presence of the divergence operator in the information field equations that fuzzy computations can be thought of as information fluid calculations.

Fuzzy Engineering is a product of at least 12 authors. It is therefore appropriate to indicate such collaboration on the cover. In fairness to the other contributors, Kosko should have been listed as the editor, rather than implied to be the sole author.



AD LINK 189

Kosko is best suited to the fuzzy-logic practitioner, or to a graduate student following the Kosko curriculum: This is not a starter book. The book does include a 3.5-inch diskette with software that runs on DOS and Windows 3.x. The programs are demos only; no source code is provided. The fuzzy cognitive map demo is only useful after thorough study of the text.

Applications of Fuzzy Logic

Applications of Fuzzy Logic, edited by Mohammad Jamshidi, Andri Titli, Lotfi Zadeh, and Serge Boverie, is a collection of 20 papers by an international panel of authors. The spectrum is wide: The systems to be measured and controlled include mobile robots, active autosuspension systems, traffic noise, cement production, automotive powertrains, locomotive slip, shuttle training aircraft, and zero-sum games.

Each paper has the familiar engineering paper outline:

- The problem is described.
- The theory that will be applied to solve the problem is described.
- The particulars about the project are delineated.
- The data are presented with graphs and text.
- Conclusions are drawn, and recommendations made, for future work.

Because the authors work independently, and produce their papers to stand alone, there is some overlap in specifying the fundamentals of fuzzy logic.

The first beef (of two) that I have with *Applications of Fuzzy Logic* is the New Registered Phrase. On the cover, it's presented as "High Machine Intelligence Quotient Systems®." In the "Preface," the text is self-praised with the sentence "This is one of the first volumes in which MIQ® is brought to the attention of the scientific community." I was disappointed that the index had no entries to what was apparently an important idea. A careful search of the text revealed the single mention of Machine Intelligence Quotient on page 103, without the ® mark. The text does not explain MIQ®, nor why it is important.

The second beef is that the four editors who collected the papers and prepared them to be published should have refined the material: You can find clinkers such as, "The essence of the developed in the work ideas allows as well come along other way, that is to design original [system] for the new plants being constructed." Practitioners are more likely to be interested in the mathematics and the graphs than in pellucid syntax, so this is more of a cavil than a beef.

Judging by the dates of the references, the papers presented here are state of the art. Practitioners in the control field will want to have a copy of this book on their shelf, but are likely to be interested only in the papers that apply to their discipline.

Applications of Fuzzy Logic does not include a diskette, and the papers contain little code. On the other hand, the authors occasionally mention the languages in which their control programs are written: Fortran, C, Matlab/Simulink, and Pascal. Computers mentioned in the papers included a DEC PDP-11/44, 20-MHz Inmos T800 Transputer, 386 laptop, and IBM RISC6000. I can only conclude that fuzzy-logic engineers are more interested in getting their jobs done, than in exploiting the hyped technology du jour.

The Design and Development of Fuzzy Logic Controllers

As I scanned *The Design and Development of Fuzzy Logic Controllers*, by Byron Miller, several simple drawings illustrated what looked like interesting low-budget projects. However, the "graphics" are the bane of this book.

On page 2, Figure 1.1 is supposed to show a dynamic system. Instead, it shows graphs of continuous and discrete data, and the graphs are mislabeled.

On page 3, Figure 1.2 is a duplicate of Figure 1.1. Again, the continuous and discrete data labels are swapped.

After an abbreviated and fairly clear survey of fuzzy logic, an input file for the FuzzyStat system is listed in its entirety. It is followed by an unintelligible and unexplained "3D graph." Several additional unintelligible input files are included, each accompanied by an equally opaque graphic.

The software-development process is diagrammed with the hoary waterfall diagram. Unfortunately, the diagram is said to demonstrate the spiral model of development.

These, and other errors, make the text unreliable and not at all credible.

Save yourself $52.00 and skip this one.

Programmer's Bookshelf

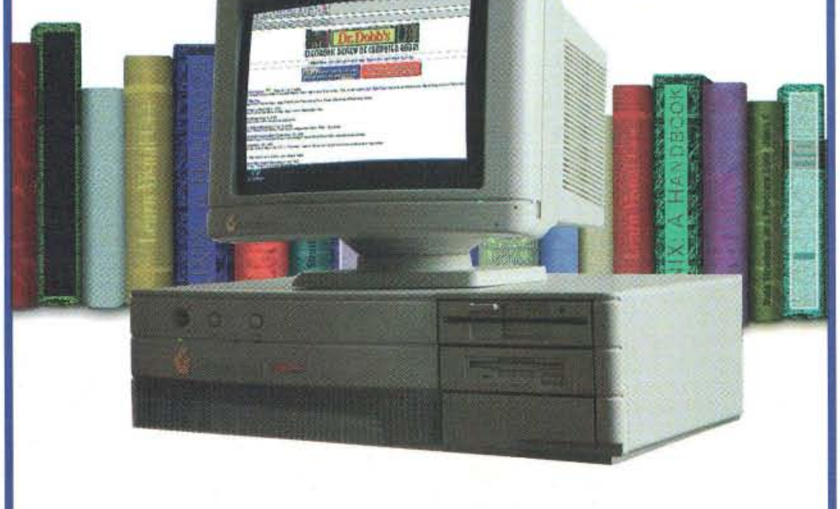
Submissions to Programmer's Bookshelf and the *Dr. Dobb's Electronic Review of Computer Books* (http://www.ddj.com/ercb/) can be sent via e-mail to editors@ddj.com or mailed to *DDJ*, 411 Borel Ave., Suite 100, San Mateo, CA 94402.

DDJ

Dr. Dobb's Electronic Review of Computer Books!

Independent reviews of technical computer books

Written BY developers FOR developers



BooksNow
The Virtual Bookstore℠

To order books in this magazine or, any book. Please call 24 hrs/365 days: **(800) BOOKS-NOW** (266-5766) or (702) 258-3338 ask for ext. 1410 or visit us on the web at http://www.BooksNow.com/Dr.Dobbs. Use Visa, M/C, or AMEX or send check or money order + $4.95 S&H ($2.50 each add'l item) to: Books Now, 6600 W. charleston Blvd. Las Vegas, NV 89102

Dr. Dobb's Journal Online Reader Service AD LINK

Ad Link www.ddj.com

Here's how to request *DDJ* advertiser information online:

1. Set your browser to **www.ddj.com**
2. Click on *DDJ* Ad Link
3. Search by product category, product name, company name, or the Ad Link number from the list below
4. Choose how you wish to receive information (e-mail, regular mail, or a phone call).

It's Fast. Try Ad Link Today!

ADVERTISERS INDEX

Ad Link#	Advertisers Name	Page #
124	Abraxas Software, Inc	112
297	Addison Wesley	117
421	Aladdin Knowledge Systems	32
*	Al Steven's Cram Course CD-ROM	75
371	Altura Software	86
143	American Cybernetics	54
422	Apex Software	104
*	Asia-Net	65
189	Black Ice Software	142
180	BlueWater Systems	85
68	Bristol Technology	23
75	Computer Associates	27
*	Computer Bowl	140
423	Coriolis Group	59
80	Date Techniques, Inc	24
*	Dell Computer	34-35
388	DevelopMentor	129
*	Diab Data	73
413	DigitalThink	113
348	Dinkumware, Ltd	101
125	Distinct Corp	91
*	*Dr. Dobb's* Electronic Books	143
*	*Dr. Dobb's* NT Internals CD-ROM	83
*	*Dr. Dobb's* CD-ROM Library	96-97
279	Dundas Software	87
*	Edinburgh Portable Compilers	111
84	FairCom	4
410	Far Point Technologies	99
*	Game Developer Conference	147
*	Gimpel Software	103
349	Greenview Data, Inc	64
605	Hendela Systems	149
374	Hitachi	107
*	ICESoft AS	48
127	IDG Books	46
*	Intel Corp	40-41
227	Intersolv	132-133
394	Iomega	50-51
238	Iona	11
*	ITCC	118
310	Kenonic Controls, Ltd	30
288	KL Group, Inc	19
*	Langner GmbH	128
90	Lead Technologies	13
419	Macro Vision	28
91	Mainsoft Corp	25
205	Mathworks	71
146	MDBS	31
*	Microsoft	16-17
*	Microsoft	49
*	Microsoft	60-61
*	Microsoft	68-69
*	Microsoft	C2-1
94	MKS	125
138	National Instruments	114
95	Neuron Data	39
*	NuMega Technologies	53
*	NuMega Technologies	67
*	On Time	81
339	Open Systems Resources	12
294	Parasoft Corp	37
*	PC Expo	139
376	Perforce	65
400	Petronio Technology Group	47
99	Phar Lap Software, Inc	77
118	Popkin Software & Systems	56
121	Premia Corp	55
405	Premia Corp	57
**	Product Debut	148-149
**	Programmer's Marketplace	150-151
128	Programmer's Paradise	14-15
134	QNX	76-77
603	Quadron	148
*	R&D Books	102
*	Raima Corp	124
424	Raleigh Group	38
315	Rational Software Corp	29
606	Redhat Software, Inc	149
215	Rogue Wave Software	63
215	Rogue Wave Software	93
203	Sandstone Technology	116
420	Scarab Software	30
408	Seapine Software	47
105	Sequiter Software	C3
309	Snowbound Software	138
*	Softel vdm	112
*	Software Research	136
286	Stingray Software	5
392	Stingray Software	7
607	Sub Systems	149
417	Sun Microsystems	8-9
195	Symantec Corp	C4
351	Techbridge Technology	33
*	technetcast.com	128
601	Tetradyne Software	148
301	Tom Sawyer	43
604	Transcender Corp	149
112	Underware	12
416	Vireo Software	101
404	Walnut Creek	135
600	WCSC	148
*	Web West '98	108
602	WIBU	148

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

* The advertiser prefers to be contacted directly

Sales Director/West
Paul Miller 650-655-4183
pmiller@mfi.com

California/Southwest
Stan Barnes 650-655-4190
sbarnes@mfi.com

David Katch 650-655-4301
dkatch@mfi.com

Northwest/Texas
Michael Beasley 650-655-4304
mbeasley@mfi.com

Tom Hudner 650-655-4323
thudner@mfi.com

Fax: 650-358-9749

Sales Director/East
Brenner Fuller 603-746-3057
bfuller@mfi.com

Mid-Atlantic/Southeast
Nicole Glazen 781-235-8568
nglazen@mfi.com

Scott Rill 781-235-8577
srill@mfi.com

New England/North Central
Linda Guyette 603-924-5971
lguyette@mfi.com

Software Careers
Hilary Walker 650-655-4193
hwalker@mfi.com

Marquita Tinio 650-655-4196
mtinio@mfi.com

Dr. Dobb's Journal
SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

SUBSCRIBE TODAY!

FOR FASTER SERVICE:

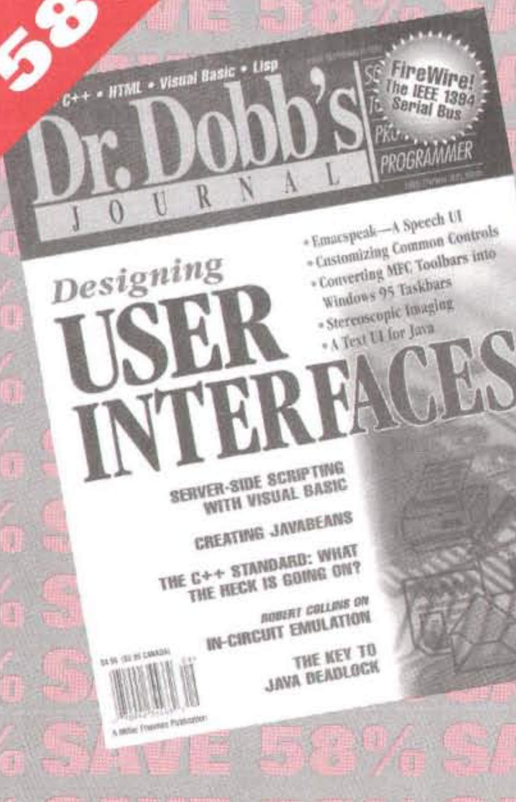
FAX: 1-(303)-661-1885

OR

VISIT OUR WEBSITE AT:

www.ddj.com

SAVE 58%



SAVINGS VOUCHER

Dr. Dobb's Journal — SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

☑ ***YES!*** I want to subscribe today! Please enter my subscription to *Dr. Dobb's Journal*. I'll get 12 monthly issues for **only $25.00** and I'll save **58% off** the newsstand rate!

Visit our Web site: http://www.ddj.com

Method of Payment Charge customers please make sure your name is clearly printed below.

☐ VISA ☐ MasterCard ☐ American Express ☐ Check or Money Order ☐ Bill Me

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Expiration Date /

Signature

Name

Company

Street/Address

City

State/Prov Zip

Country Mailcode

Email

Savings based on the full one-year rate of $59.40. International subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. (MasterCard, VISA, American Express accepted.) Canada and Mexico: $45/year (surface mail). All other foreign countries: $70/year (air courier). Canadian GST included (#R124771239). Please allow 4-6 weeks for delivery of first issue. Residents of GA, KS, & TN please add applicable sales tax.

4DSC

SAVINGS VOUCHER

Dr. Dobb's Journal — SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

☑ ***YES!*** I want to subscribe today! Please enter my subscription to Dr. Dobb's Journal. I'll get 12 monthly issues for **only $25.00** and I'll save **58% off** the newsstand rate!

Visit our Web site: http://www.ddj.com

Method of Payment Charge customers please make sure your name is clearly printed below.

☐ VISA ☐ MasterCard ☐ American Express ☐ Check or Money Order ☐ Bill Me

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 Expiration Date /

Signature

Name

Company

Street/Address

City

State/Prov Zip

Country Mailcode

Email

Savings based on the full one-year rate of $59.40. International subscriptions must be prepaid in U.S. dollars drawn on a U.S. bank. (MasterCard, VISA, American Express accepted.) Canada and Mexico: $45/year (surface mail). All other foreign countries: $70/year (air courier). Canadian GST included (#R124771239). Please allow 4-6 weeks for delivery of first issue. Residents of GA, KS, & TN please add applicable sales tax.

4DSC

PROCESS IMMEDIATELY

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO 1286 BOULDER CO

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's JOURNAL SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

PO BOX 52226
BOULDER CO 80323-2226

Please fold along this line.

Please tape closed along this edge.
(no staples)

PROCESS IMMEDIATELY

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO 1286 BOULDER CO

POSTAGE WILL BE PAID BY ADDRESSEE

Dr. Dobb's JOURNAL SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

PO BOX 52226
BOULDER CO 80323-2226

Please fold along this line.

Please tape closed along this edge.
(no staples)

Altrasoft has released OO-Browser, a high-performance, multilanguage object-oriented code browser. OO-Browser supports C, C++, Lisp and CLOS, Eiffel, Java, Objective-C, Python, and Smalltalk. OO-Browser works under Windows, MS-DOS, and many UNIX variants. Complete source code for the browser is included for local adaptation or extension. Most operations that can be applied to a single class or method can also be applied to a whole set at once. OO-Browser runs within the GNU Emacs or XEmacs editors, or as part of Altrasoft's IDE, InfoDock. It can also call out to other editors such as vi. An X Window graphical front-end is also provided. Pricing starts at $599.00.
Altrasoft
4880 Stevens Creek Boulevard, Suite 205
San Jose, CA 95129
408-243-3324
http://www.altrasoft.com/

Baltimore Technologies has released Version 3.0 of J/CRYPTO, its Java cryptography library that includes ciphers such as RSA, DES, Triple-DES, and RC4. J/CRYPTO 3.0 lets Java programs generate, process, and validate X.509 digital certificates, implements BASE64 encoding/decoding, and includes high-speed RSA performance for JDK 1.1 and higher. J/CRYPTO works with most versions of Java Virtual Machines, including 1.02 and 1.1.*.
Baltimore Technologies
International Financial Services Centre
Custom House Quay
Dublin 1
Ireland
+353 1 605 4399
http://www.baltimore.ie/

Global Checker from Uniscape is a multibyte enabling utility that helps independent software vendors reduce the time and expense of delivering multilingual products to global markets. Global Checker first scans any C/C++ type source files, then identifies and reports any code that is noncompliant under National Language Support (NLS) standards. Global Checker then suggests solutions through an extensive online help system. Global Checker is available immediately in three versions ranging in price from $400.00 to $1500.00 per unit.
Uniscape
303 Twin Dolphin Drive, Suite 510
Redwood Shores, CA 94065
650-596-1430
http://www.uni-scape.com/

Mainsoft has announced MainWin XDE 3.0, a Windows-on-UNIX platform porting technology. MainWin XDE 3.0 lets UNIX developers take advantage of Windows NT development standards—Windows 32-bit (Win32) API support, DCOM, and ActiveX—when porting to UNIX platforms. Past constraints concerning multiple data object model standards are resolved through Mainsoft's technology, allowing you to deploy enterprise applications across heterogeneous environments. MainWin XDE 3.0 is available on the SPARC Solaris 2.5.1 platform. Pricing starts at $12,000 per license.
MainSoft
1270 Oakmead Parkway, Suite 310
Sunnyvale, CA 94086
408-774-3400
http://www.mainsoft.com/

Imperial Software Technology has released VisaJ, a visual application builder for Java. VisaJ is written in Java and runs on any platform supporting JDK 1.1. VisaJ provides a point-and-click interface for building Java applications using JavaBean components, and generates pure Java code. VisaJ includes an AWT editor, layout editors, a class builder, and some essential JavaBean components. VisaJ sells for $995.00 for the first license and includes one year of support and upgrades.
Imperial Software Technology
120 Hawthorne Avenue, Suite 101
Palo Alto, CA 94301
650-688-0200
http://www.ist.co.uk/

TopSpeed has announced Clarion 4 Professional Edition, a programming environment that generates pretested code that is automatically created from two-way templates and wizards. Its data-centric environment includes a comprehensive set of objects that provide reusable services for accessing, processing, reporting, and displaying data. Clarion 4 includes TopSpeed's Database Accelerator, a database driver enhancement that incorporates caching to improve data delivery speeds. Via an alliance with Pervasive Software, Clarion 4 comes with Pervasive's Scalable SQL (two-user licenses) at no extra charge. Clarion 4 retails for $499.00.
TopSpeed Corp.
150 East Sample Road
Pompano Beach, FL 33064
954-785-4555
http://www.topspeed.com/

Objective Toolkit/X 1.0 from Stingray Software provides the capabilities of Stingray's Objective Toolkit PRO to Visual Basic programmers as an ActiveX control. Using Objective Toolkit/X, you have access to a docking form and can dock or float any MDI child form in Visual Basic. Objective Toolkit/X also includes CollectionX, an extension to Visual Basic's Collection object. Objective Toolkit/X 1.0 costs $395.00 per developer. Each copy of Objective Toolkit/X includes 60 days of technical support. Subscriptions, which include product updates and technical support for one year, are priced at $195.00.
Stingray Software
9001 Aerial Center, Suite 110
Morrisville, NC 27560
919-461-0672
http://www.stingray.com/

Global Technologies has announced its SDK for U/WIN, a UNIX-like environment for Windows-based systems with technology developed at Bell Laboratories. U/WIN comes bundled with more than 150 POSIX-compliant shell commands and utilities, and also runs the Microsoft Win32 processing mode. U/WIN is compatible with most UNIX flavors. Pricing begins at $199.00 for Windows NT, $99.00 for Windows 95.
Global Technologies Ltd.
5 West Avenue
Old Bridge, NJ 08857
609-722-0906
http://www.gtlinc.com/

Ahpah has announced its SourceAgain Professional Java decompiler. SourceAgain Pro uses flow analysis techniques to quickly and accurately translate Java class file bytecode back into Java source code equivalent to the original code. It also can augment the original class file with debugging information so that developers can step into the source code without ever needing to recompile. SourceAgain Pro runs on PCs or UNIX workstations, and works with existing IDEs. Pricing starts at $99.00.
Ahpah Software
2250 Latham Street, #21
Mountain View, CA 94040
650-960-2472
http://www.ahpah.com/

LOOX Software's LOOX 3.3 and LOOX++ 3.3 are updated versions of its flagship dynamic graphics and data visualization tools for UNIX and X/Motif developers. New features include support for animated and transparent GIF images, a scripting language for configuring complex behavioral relationships between objects, improved integration with existing Xlib code, additional predefined dynamic objects, enhanced annotation and network representation capabilities, advanced 2D charting capabilities, and extended support for hard copy output. LOOX 3.3 and LOOX++ 3.3 are available on HP-UX and Sun Solaris. A single-user license is priced at $8950.00 with no royalties or run-time fees.
LOOX Software Inc.
4962 El Camino Real, Suite 206
Los Altos, CA 94022
650-903-0942
http://www.loox.com/

TechExcel's PowerTrack 2.0 is a web-enabled defect tracker for software project management. PowerTrack tracks and manages all defects, documentation/feature requests, and development issues. New features include integration with version control software (Microsoft Visual SourceSafe), time searching, time tracking, enhanced importing and exporting, and the ability to create and use your own custom Crystal Reports within PowerTrack projects. Standard features include a scalable client/server architecture, universal ODBC support, and presentation-quality reports and graphics. PowerTrack 2.0 is priced at $389.00 per license. A fully functional evaluation copy is available at http://www.powertrack.com/.
TechExcel Corp.
3400 Mt. Diablo Boulevard, Suite 200
Lafayette, CA 94549
510-283-8930
http://www.techexcel.com/

NeoMedia Technologies has announced ADAPT/2000 Release 3.0 of its cross-platform Cobol Year 2000 tool set. ADAPT/2000 Release 3.0 provides automated source code conversion to implement IBM's recently-announced MLE (millennium language extensions) for Cobol, which provides a new syntax to automatically support windowing. ADAPT/2000 Release 3.0 allows implementation of a combination of MLE, expansion, and windowing of non-MLE date formats, and provides automatic inline bridging between date formats and generation of data conversion programs where expansion is used. Other new additions include reducing the number of code passes to catalog duplicate fields and create synonyms, greater programmer control over the automated source code changes in the remediation process, additional field-level overrides such as custom window pivot dates or conversion formats, more integrated support for embedded SQL, and numerous usability features. ADAPT/2000 starts at $20,000 for Windows NT, and $30,000 for UNIX.
NeoMedia Technologies
2201 Second Street, Suite 600
Fort Myers, FL 33901
941-337-3434
http://www.neomedia-tech.com/

Dundas Software's Ultimate ToolBox is a collection of over 200 MFC Extension Classes covering areas such as GUI, file management, communications, graphics, MAPI, and utilities. Ultimate ToolBox retails for $499.00. A yearly subscription costs an additional $249.00, and includes continuous additions of classes as they are developed.
Dundas Software
4800 Dundas Street West, Suite 500
Etobicoke, ON
Canada M9A 1B1
416-239-7472
http://www.dundas.com/

Ikonodyne has announced its Morphous Object Technology (MOT), a component-based approach for developing CORBA applications. MOT provides an IDE to create, deploy, and manage CORBA objects written in Java. MOT uses wizards to allow users to automatically build CORBA objets. CORBA objects can be converted into JavaBeans. MOT provides an administrative and monitoring tool with its object migration capabilities. MOT supports both customized and commercial ORBs, including Visigenic's Visibroker 3.0. Pricing begins at $1450.00 per seat, and includes one half day on-site training.
Ikonodyne
2680 Bayshore Parkway, Suite 106
Mountain View, CA 94043
650-938-3570
http://www.ikonodyne.com/

SuperCede for Java 2.0 is a development environment which offers complete support for JDK 1.1 and JavaBeans. SuperCede 2.0 supports compilation from Java source and bytecode as well as the dynamic loading of bytecode into running executables. SuperCede 2.0 Standard Edition costs $99.00, and includes a form-centric environment, database support, and over 60 JavaBeans. The Professional Edition adds advanced database development capabilities, a fully integrated C++ compiler, and ActiveX/JavaBean interoperability.
SuperCede Inc.
110-110th Avenue NE
Bellevue, WA 98004
425-462-7242
http://www.supercede.com/

ERTFS 1.0 from EBS is a file system for embedded applications that supports Windows 95-compatible long filenames, Windows 95-compatible FAT32 partitions, and enhanced support for PC Card ATA devices, IDE drives, LS-120 floppy disks, standard floppy disks, and RAM/ROM disks. An API is provided for file and directory management, and contiguous files are supported. ERTFS is provided in source form and may be used royalty free. The price of the complete package is $3825.00.
EBS Inc.
Box 873
Groton, MA 01450
978-448-9340
http://www.etcbin.com/

Geodesic Systems, LLC, has announced Version 3.1 of its Great Circle C/C++ debugger and memory manager for Windows 95/NT, which uses a new HTML-based GUI. The new interface makes the debugging process more visual and efficient. As with previous versions, Great Circle 3.1 continues to use garbage collection to debug and automatically manage an application's memory. Great Circle 3.1 supports Microsoft Visual C++ 4.x and 5.0, and lists for $695.00 per user license.
Geodesic Systems
414 North Orleans Street, Suite 410
Chicago, IL 60610
312-832-1221
http://www.geodesic.com/

JETS is a conformance validation suite for implementations of Java from Perennial. JETS is based on the original Java Language Specification and the Version 1.1 modifications. In addition to testing conformance, JETS enables implementation developers to generate bytecode from the test cases that can be moved to other platforms and tested for portability and functionality.
Perennial Inc.
4699 Old Ironsides Drive, Suite 210
Santa Clara, CA 95054
408-748-2900

SoftQuad Inc. introduced HoTMetaL Application Server, a server-side web application platform that uses an XML-based syntax for developing applications. HoTMetaL Application Server costs $495 per server with volume discounts.
SoftQuad Inc.
20 Eglinton Avenue West, 12th Floor
P.O. Box 2025
Toronto, ON
Canada M4R 1K8
416-544-9000
mail@softquad.com
http://www.softquad.com/

DDJ

How to rule the game market: close your office for a week.

Five days at the CGDC will make your team a lean, mean game-creating machine.

You've got to be clued in to compete—and no one knows it better than game development professionals. The Computer Game Developers' Conference gives your team the skills and strategies it takes to produce the hottest games on the market. Next-generation programming. Inspirational management. Stunning graphics and stirring audio. Brilliant marketing. Over 200 intense classes taught by the top minds in the business, covering everything from online gaming to 3D to innovative distribution tactics. Go ahead—close the office for a week. You'll come back smarter and more productive than ever before—and that lasts all year long.

(Besides, think what you'll save on overhead!)

Where Games are Born

COMPUTER Coin-Op & Console GAME DEVELOPERS CONFERENCE™

Register Today for the CGDC

MAY 4-8, 1998

Long Beach CA, Convention Center

Call 888-234-9476 or 415-905-2388

Go To www.cgdc.com

E-mail cgdc@mfi.com

DR. DOBB'S JOURNAL

PRODUCT DEBUT

Supports Windows 95 & Windows NT

SERIAL I/O

The ONLY Serial Communication Libraries, DLLs, & Tools You Will Ever Need For Windows 3.x, Windows 95, Windows NT, and MS-DOS. Accept No Less Than The Best.

http://www.wcscnet.com/home.htm

COMM-DRV/LIB™

Professional Serial I/O Libraries & DLLs for Windows 3.x, Windows 95, NT, & MSDOS

- Supports ALL languages, tools, or applications that can call the Windows API.
- **Complete source code to all libraries & DLLs.**
- Same API for Windows 3x, Windows 95, NT, & MS-DOS.
- High level Hayes compatible modem functions.
- **X, Y, Zmodem file transfers on multiple ports.**
- Asynchronous & timed callback to user functions on different serial communication events(modem signals, buffer counts, character reception, and more).
- Supports Visual C/C++, ANSI C/C++, QuickBasic, Visual Basic (MSDOS & Windows), Assembly, Access.
- Transmit data from user callback on any event.
- Extensive scanning of input character stream.
- Support most intelligent/all dumb multiport cards(Arnet, AST, Boca, Cyclades, Digiboard, GTEK, & many more).

MS-DOS Serial I/O TSRs & Utilities

- True DOS device driver that allows serial ports to be opened like files under MS-DOS & Windows.
- FOSSIL interface for all supported(Supports all BBS, Mailers, etc.[PCBoard, Wildcat, Frontdoor, Doorway, Renegade, etc.])
- Provides Int14h, Int21h, & DAM interfaces.
- Compatible with Desqview and Windows.
- Real time serial port monitoring to screen and disk.
- Mini-BBS for file transfer/Spawnable file transfer engine.
- Supports most intelligent & non-intelligent multiport cards.

COMM-DRV/VxD™

Ultra High Speed Serial I/O VxD For Windows

- Windows(95 & 3.x) DLLs with API calls into the 32bit VxD.
- MS-DOS high speed libraries for calling VxDs from a DOS box.
- All serial communication interrupts at ring 0 in 32 bit mode.
- Uses transmit & receive FIFOs(16450, 16550, 16650, etc.).
- Win32, Win16, Console, & MS-DOS apps active concurrently.
- Over 400K baud under Windows 95 & Windows 3.x!!!
- Built-in Multidrop Protocol & Nine-Bit Protocol(RS485 apps,etc).
- Compatible with Visual C/C++, Visual Basic, and many more tools.

Features Supported ByAll Products

- Remap/Change baud rates/divisor.
- Product Customization.
- Hardware/Software Flow Control
- Unlimited number of ports active concurrently.
- 8250/16450/16550 Auto detection/Up to 400 KBaud.
- 100% Port re-entrant code(Important for multitasking).
- Support all standard dumb multiport cards.

WCSC Price List

```
COMM-DRV/Lib(DOS,Win3x,Win95,NT)...$189.95
COMM-DRV/Lib(Competitive Upgrade)..$ 99.95
COMM-DRV/VxD.......................$189.95
COMM-DRV/Dos.......................$ 99.95
4Port Card(16550/8 bits)...........$170.00
4Port Card(16550/16 bits)..........$199.95
8Port Card(16550/16 bits)..........$225.00
8Port Card(16550/16 bits/Ext.Box)..$299.95
Professional Combo(Lib,VxD)........$329.95
Software Combo(Lib,VxD, Dos).......$399.95
Complete Combo(Sftwe+4Port(16550)..$499.95
```

WCSC

4425 Kingwood Dr. Suite 21, Kingwood, TX 77339
1(800)966-4832 (281)498-4832
(Fax) (281)568-3334 (BBS) (281)568-6401
Internet: sales@wcscnet.com

Visa/Mastercard/Discover/AMEX/Checks/Approved P.O. WWW: http://www.wcscnet.com/home.htm

AD LINK 600

Next Generation Software Protection

www.wibu.com

New!

WIBU-SYSTEMS introduces the future of copy protection. The new WIBU-BOX/U combines the unmatched effectiveness of hardware based copy protection with the 100% trouble free operation of the Universal Serial Bus (USB). **Call now for your free Test Kit (800) 986 6578**

ISO 9001 certified
Member of the USB Implementers Forum

GRIFFIN TECHNOLOGIES

North and South America:
Griffin Technologies, LLC
1617 St. Andrews Drive, Lawrence, KS 66047
Tel.: (785) 832-2070 · Fax.: (785) 832-8787
Email: sales@griftech.com · www.griftech.com

International
WIBU-SYSTEMS AG
Rueppurrer Strasse 54
D-76137 Karlsruhe
Tel.: +49-721-93172-0
Fax: +49-721-93172-22
Email: info@wibu.com
http://www.wibu.com

WIBU SYSTEMS
Quality the World Trusts

AD LINK 602

VxD & NT Driver Development

DFC
Driver Framework Class Library™

Create high performance VxD and NT driver modules from common source with the Tetradyne DFC library. The DFC driver wizard gets you started quickly and the DFC classes simplify common driver tasks.

Free DFC training courses (including free evaluation copy) held in San Jose, CA. Drop by www.tetradyne.com for more information.

Tetradyne SOFTWARE, INC

2542 S. Bascom Ave., Suite 206
Campbell, CA 95008
(408) 377-6367 FAX: (408) 377-6258
sales@tetradyne.com
www.tetradyne.com

AD LINK 601

COMMUNICATION TOOLS FOR

Windows NT/95 • DOS • OS/2 • QNX

Use our fast deployment tools with IBM ARTIC realtime comm cards for:

Bisync, async, frame relay, X.25, LAP-B, HDLC/SDLC, & custom protocols.

Multiple ports, multiple protocols on one card, CPU independence, & portable card development in C. Visit our web site, see for yourself & talk to us soon.

IBM ARTIC realtime communications card

Quadron®
www.quadron.com

209 East Victoria Street
Santa Barbara, CA 93101 USA
fax 805-966-7630
telephone 805-966-6424

AD LINK 603

What does it take to get Microsoft® Certified?

STUDY STUDY STUDY

TRANSCENDER® exam simulations

We Can Help!

Attaining Microsoft Certification is no easy task. But the professional and financial rewards are worth it. Transcender exam simulations give you a realistic preview of the nature and difficulty of Microsoft certification exams. Our exams show you where you need to focus your studies and give you detailed explanations of every answer to every question.

*Money Back If You Don't Pass Guarantee.**

NEW! developer pak $499

(Includes Items 1,2, & choose TWO from 3-7 listed below)

1. WinArch-I-Cert™ 2.0 *(Windows Architecture I)* - $149
2. WinArch-II-Cert™ 2.0 *(Windows Architecture II)* - $149
3. VB-Cert™ 5.0 *(Visual Basic 5.0)* - $149
4. VB-Cert™ 4.0 *(Visual Basic 4.0)* - $129
5. AccessCert™ 1.0 *(Access 2.0)* - $129
6. AccessCert™ 7.0 *(Access 7.0)* - $129
7. SQL-Cert™ 6.5 *(SQL 6.5 Implementation)* - $149

MCSE exams:
NT 4.0 (Wkstn, Srvr, Entl; NT 3.51 (Wkstn, Srvr); Win 95 & 3.1; SMS, Exchange; IIS; TCP/IP; SQL 6.5 (Impl. & Admin.); Net Ess

To order, call 615-726-8779, FAX 615-726-8884
VISA/MC/AMEX/DISC/MO/COD *Add $4 s&h ($25 outside the US)*

ONLINE ORDERING!
WWW.transcender.com

See our Web site for conditions.

Microsoft Certified Solution Provider

Transcender® Corporation
242 Louise Ave.
Nashville, TN 37203

AD LINK 604

HENDELA SYSTEM CONSULTANTS, INC.
SCANALYZER
YEAR 2000 SOLUTION SOFTWARE

Are you ready for 2000

Sizes your Y2K effort across all applications.

Analyzes your Nomad, Foxpro, COBOL, RPG, C/C++, and other text-based source code.

Gets you started fast with 1200+ date related text strings.

Remediates code with a built-in editor that positions you right to the line to be fixed.

To download a demo version or for additional product information, visit our web site:
www.scany2k.com
Call us toll-free:
1-888-SCANY2K
1-888-722-6925

AD LINK 605

Where do *you* find **116,000** professional software developers... ***monthly?*** *in*

PRODUCT DEBUT

DDJ subscribers are among the most technically advanced programmers in the field.

the secret to success is no longer a secret

www.redhat.com
The most stable, most advanced operating system

"Official" Red Hat Linux Boxed Set: $49.95
Red Hat Linux "Power Tools" & archives: $29.95
Applixware Office Suite for Linux: $99.95
Red Hat's Motif for Linux: $149
Red Hat's TriTeal CDE for Linux: $99.95
Books
Dr. Linux: $39.95
Linux Command Reference: $49.95
Maximum RPM: $34.95

redhat
1.888.redhat1 / 1.919.547.0012

AD LINK 606

7 GREAT CONTROLS!

- TE Edit Control (Advanced RTF control)
- HTML Viewer/Editor Add-on for TE
- ReportEase Plus (report writer engine)
- SpellTime DLL and dictionary
- FormPlus (form designer/filler)
- Rich Text Grid control and ChartPro

ALL PRODUCTS INCLUDE COMPLETE SOURCE CODE, ARE ROYALTY FREE AND AVAILABLE FOR 16 OR 32 BITS.

Demos: www.subsystems.com

SUBSystems

SUB SYSTEMS, INC.
11 Tiger Row, Georgetown, MA 01833
978-352-9020 Fax: 978-352-9019

AD LINK 607

DR. DOBB'S JOURNAL

MARKETPLACE

ASSEMBLY

Interactive Disassembler Pro

Interactive auto-commenting disassembler
NOW WITH STANDARD LIBRARIES RECOGNITION !
background analysis, unlimited size of input file
EXE, NE, LE, PE, LX, NLM, Binary, ROM and more
80x86, Pentium , Pentium Pro, MMX, Z80, 8051, 6502, 68K, 680X, JAVA and more - still $199 + s&h !

Fast Library Intelligent Recognition Technology relegates classical disassembler technology into prehistory. Your disassemblies suddenly come alive and familiar symbols spring in sight. Get our free evaluation from
www.datarescue.com or www.ccso.com

AD LINK 501

BOOKS

WROX PRESS

Jesse Liberty's Beginning Object-Oriented Analysis & Design with C++

Need to build an object-oriented solution? Interested in Design Patterns? If so, BOOAD explains how to design solutions for robust, maintainable and logically thought-out code.
Author, Jesse Liberty 1-861001-33-9, $34.95 At bookstores coast-to-coast. Visit www.wrox.com for a full TOC and sample chapter. 1-800-USE-WROX.

AD LINK 502

C++ LIBRARIES

html++
CGI Class Library

NEW Product

Compatible with all internet web servers
Generate interactive web pages in C++
Ideal for webifying databases
No more Perl or scripting
Automates CGI, cookies, forms, state
Win32, Win16, OS/2, DOS, Unix, Mac

FREE DEMO Call 1-800-775-1073

DC Micro Development
Tel (606) 245-4175
Fax (606) 245-9305
www.dcmicro.com

AD LINK 503

COMPRESSION

CRUSHER! Data Compression Toolkits

The best cross-platform compression libraries for Win32, Win16, DOS, OS/2, Unix, Macintosh, and embedded systems.

Robust 45-function API compresses buffers, files, archives, disk spanning, encryption, self-extr. EXE's and more.

FREE DEMO Call 1-800-775-1073

DC Micro Development
Tel (606) 245-4175
Fax (606) 245-9305
www.dcmicro.com

AD LINK 504

COMPRESSION

ZIP TOOLS

Look to Inner Media for all of your **ZIP/UNZIP** development needs:
DynaZIP™ is the industry leader for zip compatible royalty-free data compression.
Active Delivery™ is the answer when you want to add self-extracting zip file creation to your applications.
All products have a full 30-day no risk guarantee.

ROYALTY FREE!

Inner Media, Inc.
60 Plain Road, Hollis, NH 03049
(800) 962-2949
(603) 465-3216, Fax (603) 465-7195
Download our free demos!
www.innermedia.com

AD LINK 505

CONSULTING

TCOM INC.
A Device Driver Technology Company

Reduce Time to Market
Over 100 years of Driver and Embedded Expertise
Cross Platform Porting Experts

Operating Systems: WinNT•Win95•Solaris•HP-UX•AIX•Unixware IRIX•Linux•SCO•VxWorks•pSOS
Drivers: ISDN•TCP/IP•ATM•ADSL•LAN•TTY•PPP•FDDI NDIS•STREAMS•DLPI•SS7•T1/E1•X25•Many more
HW Platforms: x86•PPC•MIPS•SPARC•DSP•ISA•PCI•PCMCIA VME•PMC•CompactPCI•EISA•MicroChannel•H.100
EMail: tcom@tcom-inc.com
Web: http://www.tcom-inc.com
Voice: 973-539-1777; ask for Jim

AD LINK 506

DEVELOPMENT TOOLS

Network Software Development Toolkits

- **Win32 NDIS Framework (WinDis32)** - Supports development of Win32 applications that directly access network adapters.
- **TDI Client Samples** - Use TCP/IP from device drivers.

Printing Communications Assoc., Inc.
4201 Brunswick Court
Smyrna, GA 30080
(770) 432-4580

http//www.PCAUSA.com

AD LINK 507

Boost your Basic!

60 Day MBG!

Boost your programs into orbit! VB, QB, PB, C & Xbase libraries. Free demo disk & booklet - call now! info@teratech.com

* ProMath/VB - numerics * ProBas - DOS libs
* FinLib/VB - financial * SpellCheck/VB
* VoxLib/Pro - telephony * BarCode Tools

FREE demo!

Custom Programming: VB, C, Access, telephony, math, web CF, ASP, Java.

TeraTech tools for programmers

800-447-9120 x1336
Dept 1336, 100 Park Avenue, Suite 360, Rockville MD 20850
Int'l:+1-301-424-3903 Fax:301-762-8185 BBS:301-762-8184
www.teratech.com/ddj/

AD LINK 508

DEVELOPMENT TOOLS

Zip and Unzip from your apps with the
XCEED ZIP COMPRESSION LIBRARY

- Easy to use VBX, OCX and VCLs
- Fast, compact and reliable
- Over 40 Zip and Unzip functions!
- All 16 & 32-bit controls for $199.95

New self-extractor add-on for Xceed Zip lets your apps create customized 16/32-bit self-extracting Zip files. Only $99 with purchase of Xceed Zip!

Get the fully functional free trial version! www.xceedsoft.com/dobbs

Xceed Software 1-800-865-2626 1-514-442-2626

AD LINK 509

Code Co-op

The first fully distributed version control system

- No LAN or Central Server required.
- Synchronization through e-mail.
- Intuitive user interface.
- Easy to download (<1 MB) from our web site.

Reliable Software

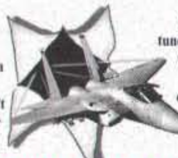
Smart Tools for Smart Programmers
http://www.relisoft.com info@relisoft.com

AD LINK 510

DOCUMENTATION TOOLS

Generate Documentation from your source code with *DocJet*!

Produce HTML, MSHelp, and MSWord documentation from comments in your code - and you won't need to change your commenting style!



You can fine-tune your output with DocJet's WYSIWYG output editor.

FREE TRIAL VERSION!
http://www.tall-tree.com
info@tall-tree.com 512-453-4909

AD LINK 511

C++ to HTML

- ***New -Version 2.1.***
- Generates documentation. directly from the source code.
- Extracts comments.
- User customized reports formats.
- HTML, WinHelp, RTF.
- *FREE* working evaluation at www.bbeesoft.com

1-888-646-1933
Bumble Bee Software
P.O. Box 2007
Westford, MA 01886
info@bbeesoft.com

AD LINK 512

EDUCATION

Netscape: Student Section

ADDRESS FOR SUCCESS

http://www.aics.edu

- Earn B.S. and M.S. in Computer Science
- DISTANCE EDUCATION
- Object oriented B.S. program
- New courses in Java, Networking, HTML, MIS
- Approved by more than 275 companies
- Follows ACM/IEEE guidelines
- Thousands of students throughout U.S.

AMERICAN INSTITUTE for COMPUTER SCIENCES

ACCREDITED MEMBER World Association of Universities and Colleges

Free catalogue ***1-800-767-AICS***
or ***http://www.aics.edu.***

FORTRAN

The SPINDRIFT Library Version 3.0

The Complete FORTRAN Programmer's Toolkit.

Create sophisticated state-of-the-art user interfaces for your FORTRAN programs. Spindrift's callable subroutines and functions let you

- Have data entry screens, dialog boxes, scrolling list boxes, pull-down menus, push buttons, help panels, etc. All with full mouse support.
- Much more! 281 subroutines and functions for keyboard, screen, mouse, DOS control, and hardware status.

Comprehensive demonstration program shows what you can do and how to do it.

Now shipping Version 3.0 with all new User Guide.

PRICE: **16 Bit Compilers - $149** **32 Bit Compilers - $289**

Spindrift Laboratories, Ltd.
8 Tanglewood Path • Galena • IL 61036 • USA
Phone: (815) 777-8240 • FAX: (815) 777-82411
E-mail: dgable@galenalink.com

AD LINK 513

GRAPHICS LIBRARIES

SciTech MGL

Now FREE with Full Source Code

SciTech MGL is a complete graphics library for Windows 95/3.1/NT & DOS that has been used to develop leading titles like *WinQuake®* and *Hexen II®*. SciTech MGL can be ported to other OS's in as few as 1000 lines of code. **Includes:** OpenGL® API support; sprite library; Game Framework; hardware triple buffering; support for stereo LCD shutter glasses; automatic detection & utilization of VGA, ModeX, VESA VBE, VBE/AF, WinG, CreateDIBSection & DirectDraw; and more. Supports standard C/C++ compilers & Borland Delphi.

For more information call **(530) 894-8400**.

Download SciTech MGL with full source code from our website at:

www.scitechsoft.com

AD LINK 514

VICTOR

Image Processing Library

Fast **BMP, TIFF, PCX, GIF, TGA, PNG, JPEG**. Adjust brightness, contrast, sharpen, create filters, resize, rotate, +more of single image, multiple images, or any image area; color reduction to optimum, specific, or std. palette; print; scan; crop, combine, compare, blend images.

DOS $199, 16-bit DLL $299, 32-bit DLL $499

Catenary Systems
314-962-7833/fax: 314-962-8037
www.catenary.com/victor

ask for free demo — src avail — visa/mc/c.o.d.

PRODUCTS WANTED

WANTED

SOFTWARE PROGRAMS

LOOking for partners in profit!

You provide the completed Small Business, SOHO, Financial or Personal Productivity software program. E-Z Legal Software will provide packaging, duplication, marketing and sales personnel to generate royalty checks for you.

With distribution in more than 6500 retail outlets, we have the experience to launch, market and sell your program at retail at no cost to you! **LINDA ZACK: 1-800-822-4566 x131**

384 S. Military Trail, Deerfield Beach, FL 33442
Phone (954) 480-8933 • Fax (954) 480-8906
http://www.e-zlegal.com — lzack@e-zlegal.com

E-Z LEGAL SOFTWARE

AD LINK 515

PROLOG TOOLS

Create Prolog Components

*Diagnose, advise, configure and plan with the Amzi!® LogicServer™ tools & libraries (DLLs) for C/C++, Java, VB, Delphi, Web Servers & more. Win NT 95 3.x & Solaris. Use ODBC, Sockets, Unicode & **new** OOP extensions.*

FREE Evaluation Version!

www.amzi.com

Amzi! inc. *Email* info@amzi.com
Call +1-513-425-8050 *Fax* 425-8025

AD LINK 516

SCRIPTING LANGUAGES

CGI scripting is a piece of cake with Winductor 2.1

- Easy and clear syntax
- Native to Windows
- Integrated debugging — can be debugged with actual POSTed data from browser
- Use e-mail, FTP, HTTP, HTML-parsing and file I/O directly from the scripts
- Script and orchestrate other Windows programs
- Easy-to-follow CGI tutorial

Download your free evaluation copy today

Winductor

www.desisoft.com Desiderata Software 2 Vernon St.#551, Framingham, MA 01701

AD LINK 517

SECURITY

Securely Sell Your Software Worldwide

Rainbow has software, hardware and Internet solutions to help developers

- ***Increase Sales***
- ***Expand Distribution***
- ***Prevent Piracy***

800-705-5552
www.rainbow.com

RAINBOW TECHNOLOGIES *Securing Software Success*

AD LINK 518

SECURITY

BugCollector Pro 2.0™

Puts Bugs In Their Place

Bug and Feature Request Tracking Database

Tracks
bugs from initial report through final resolution
Manages
features from first request through implementation
Organizes
everything you need to create a top-quality product
Sorts • Filters • Reports • Graphs • Exports
your software development data the way you want it

Nesbitt Software Corporation

Download a FREE 30-day BugCollector trial from http://www.nesbitt.com/

AD LINK 519

Reliable Software Security

Cost effective products for all levels of software protection and authentication.

- CRYPTO-BOX® brand dongles, SmartCard, PC Cards, and extension cards
- Software based metering systems
- ActiveX, OCX & VCL components

VISIT US www.marx.com ON THE WEB

MARX® SOFTWARE SECURITY 770-986-8887 FAX: 770-986-8891 1-800-627-9468
2900 Chamblee-Tucker Rd., Bldg. 9, Atlanta, GA 30341-4151

AD LINK 520

CrypKey Software Licensing System

"Software protection with **NO** hardware lock and **NO** disk key"

CrypKey is **software copy protection** that is:
- completely secure from any disk copy program
- perfect for **CD-ROM** or **INTERNET** distribution!
- cost effective, user friendly, and **100% guaranteed** to satisfy!

CrypKey can **increase your software sales:**
- upsell **options and levels** of your software
- **lease or demo** your software by runs or time
- **enable or upgrade** your customers **instantly** by **phone, fax or email!**

New! unique **Ready-To-Try** features upon install allows 1 trial period only per customer.
New! unique **Add-On** feature-add more options, levels, runs or time to existing licenses.
New! CrypKey Instant-protects in just **5 minutes** with **no source code changes.**

CrypKey is completely compatible with MS-DOS, MS-Windows 3.x, Win32s, Win95, Win NT, and manages network licenses on all Novell and Microsoft operating system based networks. CrypKey is produced by Kenonic Controls Ltd.—engineering and software since 1972.

Kenonic Controls Limited
7175 - 12th Street South East
Calgary, Alberta, Canada T2H 2S6
(403) 258-6200 fax: (403) 258-6201
INTERNET: crypkey@kenonic.com
WEB: http://www.kenonic.com/crypkey.html

AD LINK 521

TOOLS

Understand For C++

Understand Your Software...

Analyzes your C++ source code to help you quickly understand your software.

- Hypertext source & Class Browser
- Detailed Interactive Cross Reference
- Invocation, Include, Declaration Heirarchy Charts
- HTML Publishing of Code & Analysis
- WYSIWYG printing, Clipboard & Bitmap Output

Download your eval copy today.
www.scitools.com
(603) 448-6960

STI
Scientific Toolworks, Inc.
115 Etna Road Suite 18A
Lebanon, NH 03766

AD LINK 522

C and C++ DOCUMENTATION TOOLS (v. 7.0)

- **C-CALL ($69)** Graphic-tree of caller/called function hierarchy, cross-reference, file/function index.
- **C-CMT ($69)** Creates/inserts/updates comment-blocks (functions/identifiers used) for each function.
- **C-METRIC ($59)** Calculates path complexity, counts lines with comments, code, 'C' statements.
- **C-LIST ($69)** Lists and action-diagrams, or reformats source into user-selected standard formats.
- **C-REF ($69)** Creates cross-reference of local/global/define/parameter identifiers.
- **C-DOC ($199) PACKAGE** All 5 programs integrated as DOS program. <10,000 lines. C-BROWSE Windows graphic-tree viewer.
- **C-DOC Professional ($299)** DOS, Windows, OS/2, 1,000,000+ lines.
- **NEW VER 7.0! WEB HTML REPORTS!**

SOFTWARE BLACKSMITHS INC. email @ swbs.com
6064 St Ives Way, Mississauga Voice/Fax **(905) 858-4466**
ONT Canada L5N-4M1 http://www.swbs.com

AD LINK 523

Where do you find 116,000 professional software developers... *monthly?*

MARKETPLACE

MARKETPLACE

2001: The Legacy Code

Funny, the connections the mind makes. I was sitting at my keyboard, thinking that the Year 2000 problem was shaping up into quite a little horror show, when I suddenly realized that my thoughts had drifted over to the cast of the TV show "News Radio." My eyes strayed to my science fiction shelves as my fingers began to move on the keys...

The Internet radio station's manager cleared his throat. When this failed to get the attention of the staff, he tapped on his coffee cup with a pen. That worked: The cup cracked, hot coffee flowed onto his lap, and the staff looked up.

"Thank you. Oww. I called this meeting because the station is in trouble. For the past two years, since mid-1999, our advertising accounting software has been printing erroneous bills."

The announcer opened his eyes. "Kindly do not utter the words 'erroneous' and 'bill' in the same sentence."

"Sorry, Bill. And please take your feet off my desk."

"No problem, Dave. It was getting soggy anyway."

"Please pay attention; This is a pretty serious situation. It's this darned Year 2000 problem, and it's impacting the bottom line."

"I love it when you talk like that," said the sultry brunette sitting on the arm of his chair.

"The problem is legacy code," he went on.

Legacy code! The very name of eldrich horror! The entire staff shivered as though they had all spilled iced tea in their laps. The station manager turned to the Cobol mechanic who sat slouching insouciantly in a chair by the door, his legs stretched out in front of him. "You're going to have to— er, do something," the station manager told him.

The mechanic grinned, popping his gum and drawing a cigarette from the rolled-up sleeve of his t-shirt. "I'm sorry, Dave," he said. "I can't do that."

"Somehow I knew you'd say that, but occasionally you can get the machine to work by uttering the proper magical incantation."

The announcer broke in: "It's Moore's Law, Dave: 'Any sufficiently old technology is indistinguishable from magic.'"

"Well, we can't let this situation continue. I anticipated that we might have this problem. I hate to do this, but— Matthew, tell them what you've been up to."

Matthew adjusted his glasses. "I have Forth been studying. I must the problem to solve the word of unbinding speak. We the Forth must use, Luke."

"It's Dave."

"I mean Dave. Oh, darn. Now I've forgotten the word of unbinding."

"I didn't think that would work, but there are other religions."

Over the next three weeks they tried various incantations. One Indian brought an automated prayer wheel that spelled out the nine billion names of god. Didn't work. And then the station manager did what he had dreaded, and called in Monsignor Purvis and his Exorcist 2000. It got pretty ugly then.

Lightning flashed in the station manager's office and he spun around in his chair three times, spilling his coffee in his lap. The laserwriter in the outer office began spewing gross-looking green- and white-striped paper, and you wouldn't believe the language that came out of that thing.

It was Java.

Two weeks later, the station manager strode out into the outer office and tapped the receptionist's coffee cup for attention. "Well, the exorcism has worked. And it really did everything: It resolved all the Year 2000 problems, ported the code to the new server, translated two million lines of Cobol into nine billion lines of 100 percent pure Java, and morphed the whole schmear into a six-tiered intranet/extranet, replacing advertisers' snailmail addresses with e-mail addresses, not just in the database but even on their stationery and business cards. Now if we could just get it to reflect our new ad rates..."

The laserprinter spit a piece of paper on the floor. The announcer picked it up and read it in his stentorian tones. "It says, 'I'm sorry, Dave; I can't do that.'"

"Well that's just great," the station manager said, flopping down on the corner of the receptionist's desk and spilling her coffee in his lap.

"We're back where we started."

"Let me give it a try," the sultry brunette said, walking around behind the rack-mounted accounting computer. After a moment, the machine began to make moaning noises and suddenly paper began to fly out of the laser printer.

"These rates are correct," the station manager said as the brunette came out from behind the machine blowing on her fingernails. "What did you do?"

"I haven't the slightest idea," she answered enigmatically.

"Which proves my point," said the announcer. "It's incomprehensible precisely because we need it. It's Murphy's Law, Dave: 'Any sufficiently crucial technology is indistinguishable from magic.'"

Michael Swaine

Michael Swaine
editor-at-large
mswaine@swaine.com

Discover CodeBase 6.3—and get the fastest and smallest database engine available, plus xBASE file compatibility!

Query a million records in just 0.97 of a second, or add ten thousand records in just 2.17 seconds.

Discover these valuable benefits of CodeBase:

NEW

- Works with C, C++, Visual Basic, Delphi, Java and soon OLE DB
- New ActiveX data-aware controls help reduce development times
- Runs under Windows 95, NT, 3.1, CE, DOS, Mac, OS/2, Solaris, SunOS, AIX, SCO, Linux, UnixWare, DEC, Alpha, BSDI...
- FoxPro, Clipper and dBASE file support, including full multi-user compatibility
- Small library size helps applications load quickly and use less memory
- Fully scalable from stand-alone to client/server, without code changes
- Unlimited-seat client/server included
- Transaction processing and logging
- Royalty free distribution
- Full featured report writer included

DATA BASED ADVISOR Readers Choice Award

Award-Winning performance 5 years in a row!

"CodeBase gives ACT! the fast database access contact management users need"

- Michael Plasterer, Director of Development, Symantec

The Database Engine for Programmers

FREE 30 Day Test Drive

Test drive the new CodeBase 6 for 30 days with your own code. No risk. No obligation. Order yours today.

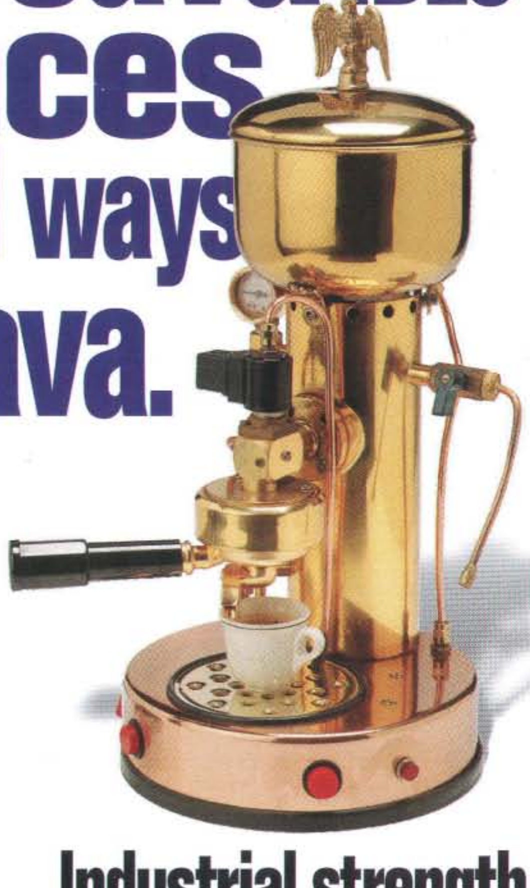
Call: 403-437-2410

SEQUITER SOFTWARE INC.
P.O. Box 783 Greenland NH 03840
Fax: 403-436-2999
Email: info@sequiter.com
Web site: www.sequiter.com

©1997 Sequiter Software, Inc. All rights reserved. CodeBase and Sequiter are trademarks of Sequiter Software Inc. All other product names are trademarks of their respective companies.

cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com •

The leader in Java™ IDEs announces two powerful ways to brew Java.



Full strength

Symantec Visual Café™ for Java
Web Development Edition

For those new to Java, and for web developers who want to push the envelope: Java applet *and* Web page creation in one fast, easy tool.

Industrial strength

Symantec Visual Café™ for Java
Professional Development Edition

For the professional Java application developer looking for a complete arsenal of state-of-the-art Java technologies in a single package.

	WDE	PDE
Comprehensive Java and Web authoring environments	✔	✔
Visual design, templates and Wizards	✔	✔
Free Netscape Communicator™	✔	✔
JDK™ 1.1 support with easy conversion of JDK 1.02 projects to JDK 1.1	✔	✔
JavaBeans components (100+) and Java Foundation Classes (JFC) included	✔	✔
Free Java book	✔	
Pure Java code to native x86 compilation (Windows version only)		✔
Incremental debugging (bytecode and native)		✔

*WDE–Web Development Edition, PDE–Professional Development Edition.

NEW VERSION 2.5

Available at: Best Buy,* * CompUSA, Computer City, Egghead, Fry's, J&R, and Micro Center

Visit our web site today
http://cafe.symantec.com/promo/dd25.html
The next generation of Java Tools

SYMANTEC.

Symantec and the Symantec logo are U.S. registered trademarks and Symantec Visual Café is a trademark of Symantec Corporation. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems in the U.S. and other countries. Microsoft, Windows, the Windows logo and Windows NT are registered trademarks of Microsoft Corporation. Other brands and products are trademarks of their respective holder/s ©1997 Symantec Corporation. All rights reserved. In Canada, call 1-800-365-8641. In Australia, call 02-9850-1000. In Europe, call 31-71-535-3111.
*For Professional Development Edition only. **WDE only.

Mac OS

cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com • cafe.symantec.com •